

MESSAGING PROGRAM, MESSAGING METHOD OF DISTRIBUTED SYSTEM, AND MESSAGING SYSTEM

Publication number: JP2002269063 (A)

Publication date: 2002-09-20

Inventor(s): ANDO TAKANOBU; YANO REI; AKIMOTO NAOTO; KURATA TOMOMI +

Applicant(s): TOSHIBA CORP +

Classification:

- international: **G06F13/00; G06F15/16; G06F15/177; G06F9/46; G06F9/54; G06F13/00; G06F15/16; G06F9/46;** (IPC1-7): G06F13/00; G06F15/16; G06F15/177

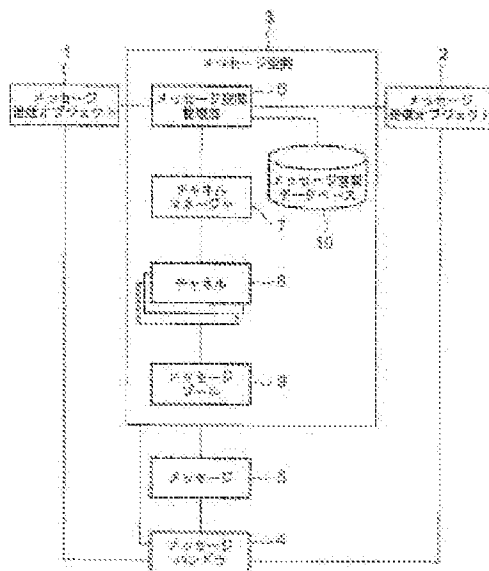
- European:

Application number: JP20010063985 20010307

Priority number(s): JP20010063985 20010307

Abstract of **JP 2002269063 (A)**

PROBLEM TO BE SOLVED: To actualize a mechanism for messaging which is usable as a reference model for a distributed system and has high flexibility in specification alteration. **SOLUTION:** This messaging program enables a computer to function as a message space managing means 6 which actualizes the management of at least one channel 8 receiving a message 5 from a message transmission side 1 and exchanging the message with a system corresponding to the nature of the message 5 and passes the received message 5 to the channel 8 corresponding to the nature of the message 5 so as to exchange messages between the message transmission side 1 and a message reception side 2 of the distributed system.



Data supplied from the **espacenet** database — Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号
特開2002-269063
(P2002-269063A)

(43) 公開日 平成14年9月20日 (2002.9.20)

(51) Int.Cl. ⁷	識別記号	F I	データポート ⁸ (参考)
G 0 6 F 15/177	6 7 6	G 0 6 F 15/177	6 7 6 Z 5 B 0 4 j 6 7 6 C 5 B 0 8 9
13/00	3 5 3	13/00	3 5 3 A
15/16	6 2 0	15/16	6 2 0 T

審査請求 未請求 請求項の数10 O L (全 26 頁)

(21) 出願番号 特願2001-63985(P2001-63985)

(22) 出願日 平成13年3月7日 (2001.3.7)

(71) 出願人 000003078

株式会社東芝

東京都港区芝浦一丁目1番1号

(72) 発明者 安東 孝信

東京都府中市東芝町1番地 株式会社東芝
府中事業所内

(72) 発明者 矢野 令

東京都府中市東芝町1番地 株式会社東芝
府中事業所内

(74) 代理人 100058479

弁理士 鈴江 武彦 (外6名)

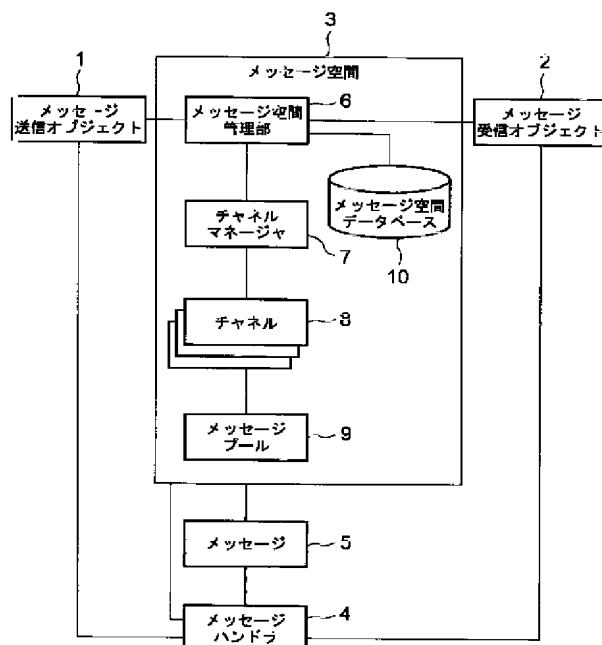
最終頁に続く

(54) 【発明の名称】 メッセージングプログラム、及び分散システムにおけるメッセージング方法、並びにメッセージングシステム

(57) 【要約】

【課題】分散システム向けの参照モデルとして利用可能であり、さらに仕様変更に対して高い柔軟性を持つメッセージングの仕組みを実現する。

【解決手段】分散システムにおけるメッセージ送信側1とメッセージ受信側2との間のメッセージ交換のためにコンピュータを、メッセージ送信側1からのメッセージ5を受け付け、メッセージ5の種類に応じた方式でメッセージ交換を制御する少なくとも一つのチャネル8の管理を実現し、受け付けたメッセージ5をそのメッセージ5の種類に応じたチャネル8に渡すメッセージ空間管理手段6として機能させるためのメッセージングプログラム。



【特許請求の範囲】

【請求項1】 分散システムにおけるメッセージ送信側とメッセージ受信側との間のメッセージ交換のために、コンピュータに、

前記メッセージ送信側からのメッセージを受け付けさせる受付手段と、

前記受付手段で受け付けた前記メッセージをそのメッセージの種類に応じたチャンネルに渡させるメッセージ空間管理手段とを有し、

前記チャンネルは、前記メッセージの種類に応じた方式で前記メッセージ交換を制御することを特徴とするメッセージングプログラム。

【請求項2】 分散システムにおけるメッセージ送信側とメッセージ受信側との間のメッセージ交換のためにコンピュータを、

前記メッセージ送信側からのメッセージを受け付け、メッセージの種類に応じた方式でメッセージ交換を制御する少なくとも一つのチャンネルの管理を実現し、受け付けたメッセージをそのメッセージの種類に応じたチャンネルに渡すメッセージ空間管理手段として機能させるためのメッセージングプログラム。

【請求項3】 請求項2記載のメッセージングプログラムにおいて、

前記メッセージ空間管理手段は、定義された共通のインタフェースとその利用手順にしたがってアクセスされることを特徴とするメッセージングプログラム。

【請求項4】 請求項2又は請求項3記載のメッセージングプログラムにおいて、

前記メッセージ空間管理手段は、前記メッセージ送信側を宛先とする送達確認メッセージを前記メッセージ受信側から受け付け、この受け付けた送達確認メッセージをその種類に応じたチャンネルに渡すことを特徴とするメッセージングプログラム。

【請求項5】 請求項1乃至請求項4のいずれか1項記載のメッセージングプログラムにおいて、

前記メッセージ空間管理手段は、メッセージ交換の履歴情報を記録し、外部からの問い合わせに応じて該当する履歴情報を提供することを特徴とするメッセージングプログラム。

【請求項6】 分散システムにおけるメッセージ送信側とメッセージ受信側との間でメッセージ交換を行うためのメッセージング方法において、

前記メッセージ送信側からメッセージを受け付ける工程と、

前記メッセージ送信側からのメッセージの種類に応じた方式でメッセージ交換を制御するチャンネルがない場合に、前記メッセージ送信側からのメッセージの種類に対応するチャンネルを生成する工程と、

前記メッセージ送信側からのメッセージをその種類に対応するチャンネルに渡す工程と、

前記メッセージ送信側からのメッセージを受けたチャンネルにより、前記メッセージ送信側からのメッセージを前記メッセージ受信側に配信する工程とを含むことを特徴とする分散システムにおけるメッセージング方法。

【請求項7】 請求項6記載の分散システムにおけるメッセージング方法において、

定義された共通のインタフェースとその利用手順にしたがってメッセージが交換されることを特徴とする分散システムにおけるメッセージング方法。

【請求項8】 請求項6又は請求項7記載の分散システムにおけるメッセージング方法において、

メッセージを受信した前記メッセージ受信側から送達確認メッセージを受け付ける工程と、

前記送達確認メッセージの種類に応じた方式でメッセージ交換を制御するチャンネルがない場合に、前記送達確認メッセージの種類に対応するチャンネルを生成する工程と、

前記送達確認メッセージをその種類に対応するチャンネルに渡す工程と、

前記送達確認メッセージを受けたチャンネルにより、前記送達確認メッセージを前記メッセージ送信側に配信する工程とを含むことを特徴とする分散システムにおけるメッセージング方法。

【請求項9】 請求項6乃至請求項8のいずれか1項記載の分散システムにおけるメッセージング方法において、

メッセージ交換の履歴情報を記録し、外部からの問い合わせに応じて該当する履歴情報を提供することを特徴とする分散システムにおけるメッセージング方法。

【請求項10】 分散システムにおけるメッセージ送信側とメッセージ受信側との間のメッセージ交換のためメッセージングシステムにおいて、

前記メッセージ送信側からのメッセージを受け付け、メッセージの種類に応じた方式でメッセージ交換を制御する少なくとも一つのチャンネルの管理を実現し、受け付けたメッセージをそのメッセージの種類に応じたチャンネルに渡すメッセージ空間管理手段を具備したことを特徴とするメッセージングシステム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、分散システムにおいてメッセージを交換するためのメッセージングプログラム、及び分散システムにおけるメッセージング方法、並びにメッセージングシステムに関する。

【0002】

【従来の技術】分散システムでは、複数の計算機に処理を分散させる。オブジェクト指向技術を利用して分散システムを構築した場合、指示命令やデータなどの情報であるメッセージが分散されているオブジェクト間でやり取りされる。

【0003】分散システムのメッセージングシステムとして、CORBAのイベントサービスや、JavaRMI (Java Remote Method Invocation)、ECJ (Event Centric for Java) などいくつかの代表的な基盤技術が存在する。

【0004】

【発明が解決しようとする課題】CORBAとは、分散システムでオブジェクト同士がメッセージを交換するためのソフトウェアである。

【0005】CORBAのイベントサービスでは、サプライヤとコンシューマとイベントチャネルからなるイベント通信モデルが提供されている。またイベント通信のスタイルとしてpush型とpull型、そしてこれらの混合型の3種類が定められている。

【0006】このように、CORBAでは基本的なメッセージングの機能や枠組みは提供されている。しかしながら、アプリケーションレベルでの送達確認機能や、イベントチャネルに対するイベントの検索機能、例えばイベントの消失などのようなエラーに関する処理などについては明確に定義されていない。また、CORBAは様々なメーカーから提供されているが、イベントサービスを利用したアプリケーションの実装方法はそれぞれによって異なる。

【0007】JavaRMIは、リモートにあるJavaオブジェクトのメソッドを他のホストのJavaVM (Java Virtual Machine) から呼び出すための分散オブジェクト技術である。これにより、異なるJavaVM上で実行中のオブジェクト同士の間でメッセージ交換が可能になる。Java RMIは基本的に同期通信に分類されるため、非同期通信の実装に直接用いることはできない。

【0008】Javaベースの非同期通信では、JMS (Java Messaging Service) がある。JMSではPeer-to-peerとPublish/Subscribeの2通りの通信モデルが提供されている。

【0009】しかしながら、JMSでも、CORBAと同様にアプリケーションレベルでの送達確認機能や通信路であるConnection/Sessionに対するメッセージの検索機能、メッセージ消失などのエラーに対する処理などは提供されない。

【0010】ECJもJavaベースのイベント通信技術である。ECJは、インターネット/イントラネット上に分散したイベント駆動型のJavaアプリケーションを構築するためのフレームワークであり、RMIを補完する非同期メッセージ交換機構を提供する。ECJではSocketを利用することで非常に軽量で高速なイベント通信を可能としている。

【0011】しかしながら、ECJでも上記と同様にアプリケーションレベルでの送達確認機能、通信路におけるメッセージの検索機能、メッセージ消失などのエラーに対する処理などは提供されていないという課題を持つ。

【0012】また上述したそれぞれのメッセージング技

術は、それぞれで実装方法が異なる。そのため、一般に各アプリケーションの実装は利用する基盤技術に依存してしまい、基盤技術の交換が大変困難である。また、基盤技術毎にメッセージングのアーキテクチャが異なるため、ある基盤技術の実装で培った経験を、他の基盤技術に対して活用しにくいという問題も生じている。

【0013】特に、従来においては、分散システム向けメッセージングのための参照モデル/フレームワークが存在しない。ここで、フレームワークとは、システムやアプリケーションを組み上げる場合に参照する大枠の構成要素の概念、あるいはこの概念を具体化したソフト基盤をいう。フレームワークを利用すればアプリケーション間の連携を標準化、統合化できるが、従来のメッセージングの技術ではこのフレームワークが存在しないため標準化、統合化が困難である。

【0014】本発明は、以上のような実情に鑑みてなされたもので、分散システム向けメッセージングのための参照モデルとして利用可能であり、さらに仕様変更に対して高い柔軟性を実現できるメッセージングプログラム、及び分散システムにおけるメッセージング方法、並びにメッセージングシステムを提供することを目的とする。

【0015】

【課題を解決するための手段】以下、本発明を実現するにあたって講じた具体的手段について説明する。

【0016】本発明は、分散システムにおけるメッセージ送信側とメッセージ受信側との間のメッセージ交換のために、コンピュータに、メッセージ送信側からのメッセージを受け付けさせる受付手段と、受付手段で受け付けたメッセージをそのメッセージの種類に応じたチャネルに渡させるメッセージ空間管理手段とを有し、チャネルは、メッセージの種類に応じた方式でメッセージ交換を制御することを特徴とするメッセージングプログラムである。

【0017】また、本発明は、分散システムにおけるメッセージ送信側とメッセージ受信側との間のメッセージ交換のためにコンピュータを、メッセージ送信側からのメッセージを受け付け、メッセージの種類に応じた方式でメッセージ交換を制御する少なくとも一つのチャネルの管理を実現し、受け付けたメッセージをそのメッセージの種類に応じたチャネルに渡すメッセージ空間管理手段として機能させるためのメッセージングプログラムである。

【0018】本発明を利用し、基盤技術毎に異なるメッセージング実装のための手段や仕組みを統一化することで、適切な参照モデル/フレームワークを提供でき、分散システム向けメッセージングのための構成を明確化できる。また、ある基盤技術の実装で培った経験を他の基盤技術で活用することが容易になる。

【0019】また、本発明で採用している参照モデル/

フレームワークは、例えばCORBA、JavaRMI、JavaとECJなどのような既存の分散システム間通信のための基盤技術を用いる場合にも実装可能である。

【0020】なお、チャネルは、チャネルマネージャによって管理され、このチャネルマネージャがさらにメッセージ空間管理手段によって管理されるとしてもよい。

【0021】本発明のメッセージ空間管理手段は、定義された共通のインタフェースとその利用手順（プロトコル）にしたがってアクセスされるように変形可能である。このような変形により、分散システムにおけるメッセージングのためのインタフェースやプロトコルを共通化できる。

【0022】また、本発明のメッセージ空間管理手段は、メッセージ送信側を宛先とする送達確認メッセージをメッセージ受信側から受け付け、この受け付けた送達確認メッセージをその種類に応じたチャネルに渡すように変形可能である。このような変形により、メッセージの送達確認機能を付加できる。

【0023】さらに、本発明のメッセージ空間管理手段は、メッセージ交換の履歴情報を記録し、外部からの問い合わせに応じて該当する履歴情報を提供するように変形可能である。このような変形により、履歴情報に基づく処理を実行してメッセージ空間に対するメッセージの検索機能、メッセージ不着信検出及びその対処機能を提供できる。

【0024】上記のようなメッセージングプログラム又はメッセージングプログラムを記録した記録媒体を用いることによって、分散システムにおけるメッセージ交換に対して簡単に上記の作用効果を実現できる。

【0025】また、本発明を利用すれば、メッセージ送信側からメッセージを受け付ける工程と、メッセージ送信側からのメッセージの種類に応じた方式でメッセージ交換を制御するチャネルがない場合に、メッセージ送信側からのメッセージの種類に対応するチャネルを生成する工程と、メッセージ送信側からのメッセージをその種類に対応するチャネルに渡す工程と、メッセージ送信側からのメッセージを受けたチャネルにより、メッセージ送信側からのメッセージをメッセージ受信側に配信する工程とを含む分散システムにおけるメッセージング方法を実施可能である。

【0026】この分散システムにおけるメッセージング方法についても上述した変形と同様の変形を行ってもよい。

【0027】また、本発明の各手段を具備したメッセージ交換システムを発明の対象としてもよい。

【0028】

【発明の実施の形態】以下、図面を参照しながら本発明の実施の形態について説明する。

【0029】（第1の実施の形態）本実施の形態においては、分散システムのためのメッセージングシステムに

ついて説明する。このメッセージングシステムは、メッセージングの仕組み、インタフェース定義、システム利用者からみた利用シーケンス、参照モデル／フレームワーク、エラーハンドリングに特徴がある。

【0030】図1は、本実施の形態に係るメッセージングシステムを利用する分散システムの構成を例示するブロック図である。

【0031】この分散システムは、メッセージ送信オブジェクト（アプリケーション）1、メッセージ受信オブジェクト（アプリケーション）2、メッセージ空間3、メッセージハンドラ4から構成されている。メッセージ送信オブジェクト1とメッセージ受信オブジェクト2との間でメッセージ5が交換される。

【0032】メッセージ送信オブジェクト1は、メッセージ5を作成し、メッセージ空間3に作成したメッセージ5を送信する。オプションで送達確認用のメッセージハンドラ4を用意し、メッセージ空間3に登録可能である。メッセージ受信オブジェクト2はメッセージ空間3からメッセージを受信する。受信方法には、受信用のメッセージハンドラ4をメッセージ空間3に登録し、それを用いてメッセージ空間3から呼び出されるようにする形式のpush型と、メッセージ受信オブジェクト2が自動的にメッセージ空間3にポーリングをかける形式のpull型の2通りの方式がある。なお、以下においては、どちらのメッセージの受け取り形式が用いられたとしても「配送する」と表記する。

【0033】メッセージ空間3は、メッセージ送信オブジェクト1から受け付けたメッセージ5をメッセージ受信オブジェクト2へ配送するための概念的な存在である。

【0034】メッセージ5はメッセージ送信オブジェクト1がメッセージ受信オブジェクト2に配送するために必要とする情報を含むデータである。

【0035】メッセージハンドラ4は、対応するメッセージ5がメッセージ空間3に受け付けられた場合に、そのメッセージ交換を要求するメッセージ送信オブジェクト1やメッセージ受信オブジェクト2をコールバックし、メッセージを受け渡すために使われるインタフェースを実装した構成要素である。

【0036】また、メッセージハンドラ4は、予定されているメッセージ5の不着が検出された際のインタフェースも備える。

【0037】メッセージ空間3はさらにメッセージ空間管理部6とチャネルマネージャ7、チャネル8、メッセージプール9、そしてメッセージ空間データベース10から構成される。

【0038】メッセージ空間管理部6は、メッセージ空間3全体に関わる処理、例えばメッセージ空間3の初期化や、メッセージ5の登録受付、チャネル8やチャネルマネージャ7の準備、メッセージハンドラ4の受付、送

信されたメッセージ5のチャンネル8への割り当てなどを行う。

【0039】メッセージ空間管理部6は、これらの処理を実現するために、メッセージ空間3内でやり取りされるメッセージ5の種類、メッセージの種類とチャンネル8の対応関係、メッセージの種類とチャンネルマネージャ7の対応関係、メッセージやメッセージの種類とメッセージ送信オブジェクト1との対応関係、メッセージやメッセージの種類とメッセージ受信オブジェクト2との対応関係などに関する情報を、メッセージ空間データベース10を用いて保持している。

【0040】メッセージ送信オブジェクト1やメッセージ受信オブジェクト2がメッセージ空間3へアクセスするためのインタフェースを提供するのもメッセージ空間管理部6である。

【0041】チャンネルマネージャ7は、チャンネル8に対する初期化や複数のチャンネル8間の制御を行う。

【0042】チャンネル8は、メッセージ5の種類毎に用意され、メッセージ空間3内でのメッセージ5の交換（やり取り）を制御する。

【0043】メッセージプール9は、メッセージ5の送信先（受け手）に一つ一つに対応する。このメッセージプール9は、メッセージ5を蓄積しておくための構成要素である。なお、メッセージプール9はオプションである。

【0044】チャンネルマネージャ7は、図2に示すように、例えばシナリオチャンネルマネージャ11などのようなチャンネル8に対して特別な制御を行う構成要素を派生する。シナリオチャンネルマネージャ11は、例えば、指定された複数種類のメッセージ5の順序を矛盾なく配送するための制御を行う。

【0045】チャンネル8は、図3に示すように、キューチャンネル12やストアチャンネル13、シンプルチャンネル14などのようなメッセージ5に対する扱いの種類に応じた構成要素を派生する。

【0046】例えば、キューチャンネル12は、図4に示すように、メッセージ送信オブジェクト1からメッセージ5を受け取ると、メッセージ5を一旦キュー15へ格納してからメッセージ受信オブジェクト2へ配送する。

【0047】ストアチャンネル13は、図5に示すように、メッセージ送信オブジェクト1からメッセージ5を受け取ると、そのメッセージ5をメッセージ受信オブジェクト2に配送するとともに、ストア16にも格納する。メッセージ受信オブジェクト2のメッセージ検索により、ストア16内のメッセージ5はメッセージ受信オブジェクト2に配信される。

【0048】シンプルチャンネル14は、図6に示すように、特にメッセージプール9は用いず、メッセージ空間3が受け付けたメッセージ5をメッセージ受信オブジェクト2にそのまま配送する。

【0049】また、どのチャンネル8もオプションで、送達確認をサポートするための機能（メッセージの送信先リスト、送信履歴、送達確認の受信履歴、これらの対応関係の監視など）を有している。

【0050】ストアチャンネル13やシンプルチャンネル14はメッセージ5の配送のために、必要に応じて配送先のメッセージハンドラ4（のリスト）を保持する。また、必要に応じて送受信したメッセージ5の履歴やその送達確認情報の履歴についても保持する。

【0051】メッセージプール9は、図7に示すように、キュー15やストア16などのようなメッセージ5を保持する方法の違いに応じた構成要素を派生する。

【0052】例えば、キュー15は、メッセージ5を受け取った順に、メッセージ受信オブジェクト2に配送する。これを実現するために、キュー15は必要に応じて配送先のメッセージハンドラ4を保持する。

【0053】ストア16は、メッセージ5をランダムアクセス可能な記憶装置へ格納する。

【0054】キュー15はさらに、優先順位付きキュー17、順序保証付きキュー18、永続キュー19などのサブ構成要素を派生する。

【0055】例えば、優先順位付きキュー17は、キュー15に貯えられているメッセージ5のうちもっとも優先順位の高いものを優先してメッセージ受信オブジェクト2へ配送する。順序保証キュー18は、指定された順序を優先してメッセージ5をメッセージ受信オブジェクト2へ配送する。永続キュー19は配送したメッセージ5の履歴情報を残し、後からも検索処理などを実行可能にする。

【0056】以下に、インタフェースの定義について説明する。

【0057】メッセージ空間3は、メッセージ送信オブジェクト1やメッセージ受信オブジェクト2に公開するインタフェースに定義されたメソッド（操作）を通して、様々な機能を提供する。

【0058】本実施の形態では、利用する通信基盤技術に依存しないインタフェースを定義する。

【0059】メッセージ空間3が提供するメソッドはその目的に応じて利用開始、メッセージの送受信、検索処理、利用終了、管理の5種に分類される。

【0060】以下に本実施の形態における定義をそれぞれ種類毎に列挙する。またメッセージハンドラが定義すべきメソッドについても定義する。

【0061】利用開始のためのメソッド(A)：

(1)送信するメッセージ5のメッセージ空間3におけるコンテキストの取得

〔定義〕メッセージ空間3に対して、メッセージ送信オブジェクト1が送信しようとしている種類のメッセージ5のコンテキストを取得するためのメソッド、コンテキストは、指定されたメッセージ5が既にメッセージ空間

3内に存在していれば、そのメッセージ5を扱っているチャンネルマネージャ7、チャンネル8、メッセージプール9の種類を示す識別子群を含む。

【0062】[シグネチャ] コンテキストを調べたいメッセージ5を引数としてもつ。返り値はコンテキストであり、メッセージ空間3内にまだ存在していなければ空の値を返す。

【0063】[動作] このメソッドが呼ばれると、メッセージ空間3は、指定されたメッセージ5がメッセージ空間管理部6で既に管理してあるかどうかを、メッセージ空間データベース10を用いて調べる。もし既に存在している場合には、そのメッセージ5を取り扱っているチャンネルマネージャ7、チャンネル8、メッセージプール9の種類を調べ、コンテキストとして返す。もし存在しない場合は空の値を返す。

【0064】(2)送信するメッセージ5の種類を登録するメソッド(その1)

[定義] メッセージ送信オブジェクト1が送信するメッセージ5をメッセージ空間3に登録するためのメソッド。メッセージ5がまだメッセージ空間内で扱われていない場合に使うメソッド。メッセージ5の種類毎に登録する。

【0065】[シグネチャ] 登録するメッセージ5とチャンネルマネージャ7、チャンネル8の種類をそれぞれ指定するための識別子と、メッセージ送信オブジェクト1を特定するための識別子をそれぞれ引数としてもつ。

【0066】[動作] このメソッドが呼ばれると、メッセージ空間3は、メッセージ5の種類とメッセージ送信オブジェクト1の識別子をメッセージ空間データベース10へ記録するとともに、適切なチャンネルマネージャ7を通して、与えられたメッセージ5のやり取りを担当するためのチャンネル8を用意し、適切な初期化を行う。また、メッセージ送信オブジェクト1の識別子をチャンネル8で記録する。さらに、チャンネル8が送達確認用のメッセージ5のためのメッセージプール9としてキュー15を用いている場合は、そのメッセージ送信オブジェクト1に対応するキュー15を用意する。

【0067】(3)送信するメッセージ5の種類を登録するメソッド(その2)

[定義] メッセージ送信オブジェクト1が送信するメッセージ5をメッセージ空間3に登録するためのメソッド。メッセージ5が既にメッセージ空間内で扱われている場合に使うメソッド、メッセージ5の種類毎に登録する。

【0068】[シグネチャ] 登録するメッセージ5とメッセージ送信オブジェクト1を特定するための識別子を引数としてもつ。

【0069】[動作] このメソッドが呼ばれると、メッセージ空間3は、メッセージ5の種類とメッセージ送信オブジェクト1の識別子をメッセージ空間データベース

10へ記録するとともに、与えられたメッセージ5をやり取りしているチャンネル8に対して、与えられたメッセージ送信オブジェクト1の識別子を記録する。さらに、チャンネル8が送達確認用のメッセージ5のためのメッセージプール9としてキュー15を用いている場合は、そのメッセージ送信オブジェクト1に対応するキュー15を用意する。

【0070】(4)送達確認をpush形式で受け取るための確認用ハンドラを登録するメソッド

[定義] メッセージ送信オブジェクト1が、送信するメッセージ5の送達確認を受けるためのメッセージハンドラ4をメッセージ空間3に登録するためのメソッド。メッセージ5の種類毎に登録する。

【0071】[シグネチャ] このメソッドは登録するメッセージハンドラ4とそれが対応するメッセージ5を指定するための識別子を引数として持つ。

【0072】[動作] このメソッドが呼ばれると、メッセージ空間3は、与えられたメッセージ5を担当するチャンネル8に、確認用ハンドラとして与えられたメッセージハンドラ4に登録する。このチャンネル8が送達確認用のメッセージ5のためのメッセージプール9としてキュー15を用いている場合は、そのメッセージ送信オブジェクト1に対応するキュー15に与えられたメッセージハンドラ4に登録する。

【0073】(5)受信するメッセージ5の種類を登録するメソッド

[定義] メッセージ受信オブジェクト2が、配送を希望するメッセージ5をメッセージ空間3に登録するためのメソッド、メッセージ5の種類毎に登録する。

【0074】[シグネチャ] このメソッドは登録するメッセージ5とメッセージ受信オブジェクト2を特定するための識別子を引数として持つ。

【0075】[動作] このメソッドが呼ばれると、メッセージ空間3は、メッセージ5の種類とメッセージ受信オブジェクト2の識別子をメッセージ空間データベース10へ記録するとともに、与えられたメッセージ5を担当するチャンネル8に、メッセージ5の配送先として指定されたメッセージ受信オブジェクト2の識別子を伝える。チャンネル8が送信用にキュー15をメッセージプール9として用いている場合は、その配送先のための対応するキュー15を作成する。

【0076】(6)メッセージ5をpush形式で受け取るためのメッセージハンドラを登録するメソッド

[定義] メッセージ受信オブジェクト2が、配送を希望するメッセージ5を受け取るためのメッセージハンドラ4をメッセージ空間3に登録するためのメソッド。メッセージ5の種類毎に登録する。

【0077】[シグネチャ] このメソッドは登録するメッセージハンドラ4とそれが対応するメッセージ5を指定するための識別子を引数として持つ。

【0078】[動作] このメソッドが呼ばれると、メッセージ空間3は、メッセージ5の種類とメッセージ受信オブジェクト2の識別子をメッセージ空間データベース10へ記録するとともに、与えられたメッセージ5を担当するチャンネル8に、与えられたメッセージハンドラ4を登録する。チャンネル8が送信用に、キュー15をメッセージプール9として用いている場合は、その配送先のための対応するキュー15を作成し、与えられたメッセージハンドラ4を登録する。

【0079】(7)メッセージ5の不着検出のための属性を設定するメソッド

[定義] メッセージ5の不着検出のため、メッセージ5が送信される予定をメッセージハンドラ4に設定するためのメソッド。

【0080】[シグネチャ] 対象となるメッセージ5と、その送信に関する予定を表すオブジェクト。このオブジェクトには、そのメッセージ5に関する定期性や予定時刻などが設定されている。

【0081】[動作] 予定を表すオブジェクトの情報に従って、メッセージが未着の場合にメッセージ空間3の検索機能(後述)を用いて、所望のメッセージ5の状況を調べる。未着の場合は対応するメッセージハンドラ4の不着検出時に呼ばれるメソッド(後述)を呼び出す。

【0082】メッセージ送受信のためのメソッド(B):
(1)メッセージ5を送信するためのメソッド

[定義] メッセージ空間3に対して、メッセージ送信オブジェクト1がメッセージ5を送信するためのメソッド。

【0083】[シグネチャ] 送信するメッセージ5を引数としてもつ。

【0084】[動作] このメソッドが呼ばれると、メッセージ空間管理部6は、与えられたメッセージ5の種類とそれを特定する識別子、それが送信された時間をメッセージ空間データベース10へ記録するとともに、与えられたメッセージ5を、それを担当するチャンネル8に渡す。チャンネル8は自分の種類に応じた方式でメッセージ配信を行う。

【0085】(2)送達確認を送信するためのメソッド

[定義] メッセージ空間3に対して、メッセージ受信オブジェクト2が、受け取ったメッセージ5の送達確認を送信するためのメソッド。

【0086】[シグネチャ] 送信する送達確認を引数として持つ。

【0087】[動作] このメソッドが呼ばれると、メッセージ空間管理部6は、与えられた送達確認のもととなるメッセージ5を担当するチャンネル8に渡す。チャンネル8は自分の種類に応じた方式で送達確認を配信する。

【0088】(3)送達確認をpull形式で受信するためのメソッド

[定義] メッセージ空間3に対して、メッセージ送信オ

ブジェクト1がメッセージ5の送達確認をpull形式で受信するためのメソッド。

【0089】[シグネチャ] 送信したメッセージ5を引数としてもつ、オプションで、時間やメッセージ5を特定するための識別子を引数として渡す場合もある。返り値は送達確認。

【0090】[動作] このメソッドが呼ばれると、メッセージ空間管理部6は与えられたメッセージ5を扱うチャンネル8に対して、与えられたメッセージ5に対応する送達確認を要求する。対応する送達確認がチャンネル8に既に届けられていれば、それを返す。まだ届けられていなければ、空を返す。チャンネル8の詳細な動作はその種類に依存する。時間やメッセージ5を特定するための識別子が渡された場合は、その値に応じたメッセージ5を配送する。

【0091】(4)メッセージ5をpull形式で受信するためのメソッド

[定義] メッセージ空間3に対して、メッセージ受信オブジェクト2がメッセージ5をpull形式で受信するためのメソッド。

【0092】[シグネチャ] 必要に応じて、メッセージ受信オブジェクト2を特定するための識別子を引数としてもつ。オプションで、時間やメッセージ5を特定するための識別子を引数として渡す場合もある。返り値はメッセージ5。

【0093】[動作] このメソッドが呼ばれると、メッセージ空間管理部6は与えられたメッセージ5を扱うチャンネル8に対して、メッセージ5の要求を伝える。チャンネル8はその種類に応じた方式でメッセージ受信オブジェクト2にメッセージを配送する。時間やメッセージ5を特定するための識別子が渡された場合は、その値に応じたメッセージ5を配送する。

【0094】メッセージ検索のためのメソッド(C):

(1)メッセージ空間3でやり取りされているメッセージ5の種類の一覧を返すメソッド

[定義] メッセージ空間3に対して、その利用者(メッセージ送信オブジェクト1及びメッセージ受信オブジェクト2)が、現在メッセージ空間3内でやり取りされているメッセージ5の種類の一覧を問い合わせるためのメソッド。

【0095】[シグネチャ] 問い合わせる際の条件式を引数としてもつ。この条件式は省略可能。返り値は、現在メッセージ空間3内でやり取りされている。条件式に該当するメッセージ5の一覧。条件式が省略されている場合は、全メッセージ5の一覧。

【0096】[動作] このメソッドが呼ばれると、メッセージ空間管理部6は、現在登録されているメッセージ5のうち条件式に合致するものをメッセージ空間データベース3に問い合わせ、一覧として返す。条件式が省略されている場合は、現在やり取りされている全メッセー

ジ5の一覧を作成し、返す。

【0097】(2)指定されたメッセージ5の送信オブジェクト一覧を返すメソッド

〔定義〕メッセージ空間3に対して、その利用者（メッセージ送信オブジェクト1及びメッセージ受信オブジェクト2）が、現在メッセージ空間3内でやり取りされている任意のメッセージ5の送信オブジェクトの一覧を問い合わせるためのメソッド。

【0098】〔シグネチャ〕送信オブジェクト一覧を調べたいメッセージ5を引数としてもつ。返り値は、現在のメッセージ空間3に指定されたメッセージ5を送信するメッセージ送信オブジェクト1の一覧。

【0099】〔動作〕このメソッドが呼ばれると、メッセージ空間管理部6は、与えられたメッセージ5のメッセージ送信オブジェクト1をメッセージ空間データベース10に問い合わせ、一覧として返す。

【0100】(3)指定されたメッセージ5の受信オブジェクト一覧を返すメソッド

〔定義〕メッセージ空間3に対して、その利用者（メッセージ送信オブジェクト1及びメッセージ受信オブジェクト2）が、現在メッセージ空間3内でやり取りされている任意のメッセージ5の受信オブジェクトの一覧を問い合わせるためのメソッド。

【0101】〔シグネチャ〕受信オブジェクト一覧を調べたいメッセージ5を引数としてもつ。返り値は、現在のメッセージ空間3に指定されたメッセージ5を受信するメッセージ受信オブジェクト2の一覧。

【0102】〔動作〕このメソッドが呼ばれると、メッセージ空間管理部6は、与えられたメッセージ5のメッセージ受信オブジェクト2をメッセージ空間データベース10に問い合わせ、一覧として返す。

【0103】(4)指定されたメッセージ5の履歴一覧を返すメソッド

〔定義〕メッセージ空間3に対して、その利用者（メッセージ送信オブジェクト1およびメッセージ受信オブジェクト2）が、これまでにメッセージ空間3内でやり取りされていた任意のメッセージ5の一覧を問い合わせるためのメソッド。

【0104】〔シグネチャ〕履歴一覧を調べたいメッセージ5を引数としてもつ。返り値は、これまでにメッセージ空間3でやり取りされ、指定されたメッセージ5の一覧。

【0105】〔動作〕このメソッドが呼ばれると、メッセージ空間管理部6は、与えられたメッセージ5の履歴情報をメッセージ空間データベース10に問い合わせ、一覧として返す。

【0106】(5)指定されたメッセージ5が指定された時間以降にやり取りされているかどうかを調べるためのメソッド。

【0107】〔定義〕メッセージ空間3に対して、その

利用者（メッセージ送信オブジェクト1及びメッセージ受信オブジェクト2）が、あるメッセージ5について指定した時間以降にやり取りされているかどうかを調べるためのメソッド。

【0108】〔シグネチャ〕調べたいメッセージ5と指定する時間を引数として持つ。返り値は、やり取りされていたかどうかを示す2値。

【0109】〔動作〕このメソッドが呼ばれると、メッセージ空間管理部6は、与えられたメッセージ5が与えられた時間以降で、メッセージ空間3内でやり取りされた記録があるかどうかを、メッセージ空間データベース10に問い合わせて調べ、その結果を返す。

【0110】利用終了のためのメソッド (D) :

(1)メッセージ空間3からの離脱を実施するメソッド

〔定義〕メッセージ空間3に対して、その利用者（メッセージ送信オブジェクト1及びメッセージ受信オブジェクト2）が、それ以降、メッセージ5をやり取りしないことを宣言するためのメソッド。

【0111】〔シグネチャ〕登録していたメッセージ5とその利用者（メッセージ送信オブジェクト1およびメッセージ受信オブジェクト2）を特定するための識別子を引数としてもつ。

【0112】〔動作〕このメソッドが呼ばれると、メッセージ空間管理部6は、メッセージ空間データベース10における与えられた利用者の識別子に関する情報を削除するとともに、与えられたメッセージ5を扱っていたチャネル8に対して、与えられた識別子の利用者が、以後そのメッセージ5を扱わないこととするための処理を行う。キュー15を用いていた場合は、そのキュー15を始末する。

【0113】メッセージハンドラのためのメソッド (E) :

(1)メッセージ5が渡された際に呼び出されるメソッド

〔定義〕メッセージ5がメッセージ空間3へ送信された際に、push型でそのメッセージ5を受け取る場合に利用されるメッセージハンドラ4が実装するメソッド。

【0114】〔シグネチャ〕メッセージ5を引数にもつ。

【0115】〔動作〕このメソッドの動作はアプリケーション開発者に委ねられる。

【0116】(2)メッセージ5の不着が検出された際に呼び出されるメソッド

〔定義〕メッセージ5の不着が検出された際に呼び出される、メッセージハンドラ4が実装するメソッド。

【0117】〔シグネチャ〕エラーメッセージを含むオブジェクトを引数にもつ。

【0118】〔動作〕このメソッドの動作はアプリケーション開発者に委ねられる。

【0119】以下に、システム利用者（メッセージ送信オブジェクト1及びメッセージ受信オブジェクト2）か

らの利用シーケンスについて説明する。

【0120】図8は、システム利用者からみた本実施の形態におけるメッセージ空間3を利用するための流れ図である。この図8に示すように、初期化(ST801)、運用(ST802)、終了処理(ST803)の順に実行される。

【0121】以下に、メッセージ送信オブジェクト1からの利用シーケンスについて説明する。

【0122】図9は、メッセージ送信オブジェクト1からみた初期化(ST801)に関する流れ図である。この図9に示すように、コンテキストの取得(ST901)、登録(ST902)、送達確認用のメッセージハンドラの登録(ST903)の順に実行される。送達確認用のメッセージハンドラの登録(ST903)はオプションである。

【0123】コンテキストの取得(ST901)では上記メソッド(A)-(1)を用い、登録(ST902)では上記メソッド(A)-(2)、(A)-(3)を用い、送達確認用のメッセージハンドラの登録(ST903)では上記メソッド(A)-(4)を用いる。

【0124】図10は、メッセージ送信オブジェクト1からみた運用(ST802)におけるメッセージ5のやり取りに関する流れ図である。この図10に示すように、メッセージの送信(ST1001)、送達確認の受信(ST1002)の順に実行される。送達確認の受信(ST1002)はオプションである。

【0125】メッセージの送信(ST1001)では上記メソッド(B)-(1)を用い、送達確認の受信(ST1001)では、pull形式で利用する場合には上記メソッド(B)-(3)を用い、push形式で利用する場合には上記メソッド(A)-(4)で登録したメッセージハンドラで実装する上記メソッド(E)-(1)が用いられる。

【0126】メッセージ送信オブジェクト1からみた運用(ST802)におけるメッセージ検索については、上記で定義した処理(C)の単一の処理となるので、特に流れ図は定義しない。

【0127】メッセージ送信オブジェクト1からみた終了処理(ST803)では、上記で定義したメソッド(D)-(1)を用いる単一の処理となるので、特に流れ図は定義しない。

【0128】以下に、メッセージ受信オブジェクト2からの利用シーケンスについて説明する。

【0129】図11は、メッセージ受信オブジェクト2からみた初期化(ST801)に関する流れ図である。この図11に示すように、登録(ST1101)、メッセージハンドラの登録(ST1102)の順に実行される。メッセージハンドラの登録(ST1102)はオプションである。

【0130】登録(ST1101)では上記メソッド(A)-(5)を用い、メッセージハンドラの登録(ST1102)では上記メソッド(A)-(6)を用いる。

【0131】図12は、メッセージ受信オブジェクト2からみた運用(ST802)におけるメッセージ5のやり取りに関する流れ図である。この図12に示すように、受信

(ST1201)、送達確認の送信(ST1202)の順に実行される。送達確認の送信(ST1202)はオプションである。

【0132】受信(ST1201)では、pull形式で利用する場合には上記メソッド(B)-(4)を用い、push形式で利用する場合には上記メソッド(A)-(6)で登録したメッセージハンドラで実装する上記メソッド(E)-(1)が用いられる。送達確認の送信(ST1202)では上記メソッド(B)-(2)を用いる。

【0133】メッセージ受信オブジェクト2からみた運用(ST802)におけるメッセージ検索については上記で定義した処理(C)の単一の処理となるので、特に流れ図は定義しない。

【0134】メッセージ受信オブジェクト2からみた終了処理(ST803)では、上記で定義したメソッド(D)-(1)を用いる単一の処理となるので、特に流れ図は定義しない。

【0135】以下に、本実施の形態に係るメッセージングシステムの仕組みを実現するための参照モデルについて説明する。以下で説明する分散システムのためのメッセージングシステムの参照モデルは、上述した構成やインタフェース定義を前提としている。

【0136】図13は、メッセージングシステムのクラス図である。表記にはUMLを用いている。このクラス図を構成するクラスについてそれぞれ説明する。

【0137】メッセージ送信部20は、本実施の形態に係るメッセージングシステムにおける参照モデルを利用し、メッセージ5を送信するアプリケーションである。このメッセージ送信部20は、上記図1のメッセージ送信オブジェクト1に相当する。

【0138】メッセージ受信部21は、本実施の形態に係るメッセージングシステムにおける参照モデルを利用し、メッセージ5を受信するアプリケーションである。このメッセージ受信部21は、上記図1のメッセージ送信オブジェクト2に相当する。

【0139】メッセージ検索インタフェース22は、メッセージ空間3におけるメッセージ検索を実現するためのインタフェース定義であり、上記メソッド(C)を定義する。

【0140】送信側インタフェース23は、メッセージ送信部20が利用するメッセージ空間3に対するインタフェース定義である。

【0141】受信側インタフェース24は、メッセージ受信部21が利用するメッセージ空間3に対するインタフェース定義である。

【0142】送信側管理部25は、メッセージ送信部20がメッセージ空間3とやり取りするための送信側の様々な処理を行うクラスであり、送達確認の不着調査も担当する。

【0143】受信側管理部26は、メッセージ受信部21がメッセージ空間3とやり取りするための受信側の様

々な処理を行うクラスであり、メッセージの不着調査も担当する。

【0144】メッセージ空間管理部27は、メッセージ空間3の中心となるクラスである。このメッセージ空間管理部27は、送信側管理部25や受信側管理部26とのやり取りを行い、初期化処理やメッセージ送受信の際のデータベースへの記録保持や適切なチャネルの振り分け、そして終了処理などを行う。このメッセージ空間管理部27は、上記図1のメッセージ空間管理部6に相当する。

【0145】メッセージ空間データベース28は、メッセージ空間3が保持する各種データを保存するためのクラスである。保持するデータには、例えばチャネル30（このチャネルが利用するメッセージプール31やチャネルマネージャ29なども含む）の種類とメッセージ送信部20及びメッセージ受信部21の対応関係を示すデータ、メッセージ5毎の履歴情報などがある。メッセージ空間データベース28は、上記図1のメッセージ空間データベース10に相当する。

【0146】チャネルマネージャ29は、少なくとも一つのチャネル30を管理するためのクラスである。このチャネルマネージャ29は、上記図1のチャネルマネージャ7に相当する。

【0147】チャネル30は、メッセージの種類毎に用意されてメッセージのやり取りを管理するクラスである。チャネル30は、送達確認処理を実現するためのメッセージと送達確認の対応関係に関する情報を保持する。このチャネル30は、上記図1のチャネル8に相当する。

【0148】メッセージプール31は、メッセージ5をメッセージ空間内に蓄えておくためのクラスである。このメッセージプール31は、上記図1のメッセージプール9に相当する。

【0149】メッセージ32は、本実施の形態においてやり取りされるメッセージ5を示すクラスである。メッセージ32は、上記図1のメッセージ5に相当する。

【0150】送達確認33は、メッセージ5の送達確認に利用される情報を表すクラスであり、メッセージ32を継承する。

【0151】コンテキスト34は、あるメッセージ5のやり取りが、メッセージ空間3内でどのように実現されているかを表すクラスである。

【0152】送達確認用メッセージハンドラ35は、送達確認33をpush型で受け取るためのクラスである。この送達確認用メッセージハンドラ35は、上記図1のメッセージハンドラ4に相当する。

【0153】メッセージハンドラ36は、メッセージ32をpush型で受け取るためのクラスである。このメッセージハンドラ36は、上記図1のメッセージハンドラ4に相当する。

【0154】これらのクラスのうち、チャネルマネージャ29、チャネル30、メッセージプール31は、上記図2、3、7に示したように派生して利用されることが考えられる。この派生の例を図14～図16に示す。

【0155】また、代表的なシナリオに対するシーケンス図を示す。ただし、チャネル30から先の動作はチャネル30の種類／性質によって異なるため、それ以降の詳細なシーケンスはここでは省略する。

【0156】図17は、メッセージ送信部20による初期化のシーケンス図である。図18は、メッセージ受信部21による初期化のシーケンス図である。図19は、メッセージ送信部20によるメッセージ送信のシーケンス図である。図20は、メッセージ受信部21によるpush型のメッセージ受信のシーケンス図である。図21は、メッセージ受信部21によるpull型のメッセージ受信のシーケンス図である。図22は、メッセージ送信部20による終了処理のシーケンス図である。なお、メッセージ受信部21による終了処理のシーケンス図も上記図22と同様である。

【0157】以下に、分散システムのメッセージ交換でエラーが発生した場合のエラーハンドリングについて説明する。

【0158】図23は、上記のようなメッセージングシステムにおいて考えられるエラーの発生箇所、種類を示すブロック図である。

【0159】メッセージ送信オブジェクト1における送信エラー、シリアライズの失敗などのようなエラーE1には、送信側管理部25が対処の責任を負うとする。

【0160】メッセージ空間3における復元の失敗、受信エラーなどのようなエラーE2、送信エラー、シリアライズの失敗などのようなエラーE3、メッセージプールのあふれなどのようなエラーE5には、メッセージ空間管理部27が対処の責任を負うとする。

【0161】メッセージ受信オブジェクト2における復元の失敗、受信エラーなどのようなエラーE4には、受信側管理部26が対処の責任を負うとする。

【0162】なお、具体的な対処方法はエラー毎に定める。

【0163】図24は、本実施の形態に係るメッセージングを模式的に表したブロック図である。

【0164】この図24では、メッセージの伝送及び送達確認メッセージの伝送にキューを用い、どちらもpush型の配送をする構成を例に表している。

【0165】メッセージ送信オブジェクト46は、メッセージ空間インタフェース47を通してメッセージ空間管理部6にメッセージを送信する。

【0166】メッセージ空間管理部6は、必要な情報を記録すると同時に、適切なチャネルマネージャ7に依頼してキューチャネル12を生成し（既にこのメッセージに対応するキューチャネル12が存在している場合、こ

のプロセスは省略される)、キューチャネル12にメッセージを渡す。

【0167】キューチャネル12は、配信先として登録してあるメッセージ受信オブジェクト48の分だけメッセージを複製する。そして、キューチャネル12は、各メッセージ受信オブジェクト48用のキュー49にメッセージを渡し、これと同時に図25に示す形式で各宛先へメッセージを配信したことを記録する。

【0168】キュー49は、受け取ったメッセージを自分に登録されているメッセージハンドラ50を通してメッセージ受信オブジェクト48に送る。

【0169】メッセージ受信オブジェクト48は、メッセージを受け取ったことが確認できると、送達確認メッセージをメッセージ空間インタフェース47経由でメッセージ空間管理部6へ送る。

【0170】メッセージ空間管理部6は、送達確認メッセージの送信原因となったメッセージを扱ったキューチャネル12に受け取った送達確認メッセージを渡す。

【0171】キューチャネル12では、送達確認メッセージを受け取ったことを図25に示す形式で記録し、送達確認メッセージの送信原因となったメッセージ送信オブジェクト46に対応する送達確認用のキュー51に、送達確認メッセージを渡す。これを受け取った送達確認用キュー51は登録されている送達確認用メッセージハンドラ52を通してメッセージ送信オブジェクト46に送達確認がなされたことを伝える。

【0172】図26は、上記図24に示すメッセージングシステムを上記図13のクラス図に基づいて配置図を兼ねたオブジェクト図として表した図である。

【0173】部分53が本実施の形態におけるメッセージングの仕組みに相当する。

【0174】この図26では送信側54、受信側55、メッセージ空間3がそれぞれ分けて記載されているが、メッセージ空間3の実装の実体がどこにあるかは特に限定されない。すなわち、送信側54や受信側55とは異なるハードウェア上で実装される場合も考えられるし、送信側54や受信側55の一方もしくは双方と同じハードウェア上で実装されてもよい。図26中のアプリ側通信実装部56、57とメッセージ空間通信実装部58、59は実装の際に利用する分散通信技術に依存する部分である。すなわち、アプリ側通信実装部56、57とメッセージ空間通信実装部58、59は、利用する分散通信技術に応じて生成される。

【0175】また、メッセージ空間3を管理するのは基本的にメッセージ空間管理部27であるが、負荷分散やフォールトトレランスの観点から、メッセージ空間管理部27の請け負う仕事を分散させることも考えられる。この様子を図27に示す。ここではメッセージ空間管理部27の仕事を分散させるメッセージ空間サブ管理部60を備えている。サーバの多重化技術には、既存のクラ

スタリングなどの技術を用いる。

【0176】以上説明した本実施の形態に係る分散システムのためのメッセージングシステムにおいては、インタフェース定義とそれへのアクセス方法を定義した参照モデルを採用している。

【0177】これにより、この定義されたインタフェースとそれへのアクセス方法を利用することで、実装基盤などに依存しない統一的なメッセージングが可能になる。

【0178】また、統一的なメッセージングを可能とする参照モデルを定義したことで、実装技術に関係なく利用可能なモデルを明確にすることができる。このため、ソフトウェアの共通化／再利用や保守／管理が容易になり、仕様変更の波及範囲を明確化できる。また、メッセージを交換するための仕組みや機能、実装方法を統一化することにより、ある基盤技術の実装で培った経験を他の基盤技術で容易に活用できる。

【0179】また、本実施の形態に係るメッセージングシステムにおいては、既存の実装技術では提供されないメッセージの送達確認機能やメッセージ空間に対する検索機能、メッセージ不着検出およびその対処機能を提供できる。

【0180】また、本実施の形態に係るメッセージングシステムは、階層型の構造をしている。これにより、例えば業務の細部の仕様変更、新しい業務の追加、ハードウェア構成の仕様変更などのような様々な規模の仕様変更に対応できる。

【0181】特に、例えばCORBA、JavaRMI、Java及びEJBなどのような既存の分散システム間通信の実装技術に本実施の形態に係るメッセージングシステムを採用することにより、内部構造に互換性を持たせることができる。これにより、既存の分散システム間通信の実装技術に関わるソフトウェア部品の共通化／再利用や保守／管理が容易になり、仕様変更の波及範囲の明確化できる。

【0182】なお、本実施の形態で説明したメッセージングシステムの機能は、コンピュータに実行させることのできるプログラムとし、例えば磁気ディスク（フロッピー（登録商標）ディスク、ハードディスク等）、光ディスク（CD-ROM、DVD等）、半導体メモリなどの記録媒体に書き込んでコンピュータに適用可能である。また、このプログラムを通信媒体により伝送してコンピュータに適用することも可能である。上記機能を実現するコンピュータは、このプログラムを読み込み、プログラムによって動作が制御されることにより、上述した処理を実行する。

【0183】（第2の実施の形態）本実施の形態においては、CORBAのための参照モデルのマッピング手法について説明する。

【0184】図28は、分散通信技術にCORBAを使用した場合のオブジェクト図である。この図28は、上記図

26の分散通信技術にCORBAを用いた場合のpush型のオブジェクト図である。

【0185】また、図29はpull型のオブジェクト図である。なお、分散通信技術によらない部分(図26の構成要素20、23、61、25、21、36、24、26、27、62、63)は省略している。

【0186】通信側54と受信側55とのイベントの配送にはCORBAのイベントサービスを用いる。通常、CORBAのイベントサービスではイベントの送信側オブジェクトをサプライヤと受信側オブジェクトをコンシューマと呼び、イベントチャンネルにサプライヤとコンシューマを登録することによってイベントチャンネルが通信を仲介する。

【0187】Push型のイベント通信の場合、イベント送信者であるPushSupplier66はCORBAイベントサービスのEventChanel67、ProxyPushConsumer68、SupplierAdmin69と通信し、PushSupplier66がイベント送信者であることを登録する。

【0188】同様に、イベント受信者であるPushConsumer70はEventChanel67、ProxyPushSupplier71、ConsumerAdmin72と通信し、PushConsumer70がイベント受信者であることを登録する。以後、PushSupplier66がProxyPushConsumer68にイベントを登録(push)するとPushConsumer70はProxyPushSupplier71からイベントを受け取ることができる。

【0189】さらに、イベント配送にCORBAのイベントサービスを使用するのみならず、送信側54とメッセージ空間3、受信側55とメッセージ空間3の通信を実現する必要がある。送信側54のアプリ側通信実装部56とメッセージ空間通信実装部58は、アプリ側通信実装部スケルトン76、アプリ側通信実装部スタブ77、メッセージ空間通信実装部スタブ78及びメッセージ空間通信実装部スケルトン79によってCORBAによる通信を行う。

【0190】同様に、受信側55のアプリ側通信実装部57とメッセージ空間通信実装部59は、アプリ側通信実装部スケルトン82、アプリ側通信実装部スタブ83、メッセージ空間通信実装部スタブ84およびメッセージ空間通信実装部スケルトン85によってCORBAによる通信を行う。

【0191】以上のように、本実施の形態においては、既存の分散システム間通信の実装技術として代表的なCORBAのための参照モデルのマッピング手法を示した。これにより、CORBAを利用したシステムの開発が容易になる。また、上記第1の実施の形態で説明したメッセージングシステムと同様のモデルを利用することで、例えばJavaRMIやJava及びEJCなどのような他の分散システム間通信の実装技術との間で内部構造に互換性を持たせることもできる。したがって、ソフトウェア部品の共通化/再利用や保守/管理が容易になり、仕様変更の波及範囲

を明確化できる。

【0192】(第3の実施の形態)本実施の形態においては、JavaRMIのための参照モデルのマッピング手法について説明する。

【0193】図30は、分散通信技術にJavaRMIを使用した場合の模式図である。この図30は、上記図26中における分散通信技術に依存する部分をJavaRMIで実現している。分散通信技術に依存しない部分についての具体的な説明を省略する。

【0194】分散通信を行うために、アプリケーション開発者は以下の操作を行う。

【0195】まず、上記図26のアプリ側通信実装部56、57とメッセージ空間通信実装部58、59の各々について、rmicコンパイラを用いてアプリ側通信実装部_Skel90、91、アプリ側通信実装部_Stub92、93、メッセージ空間通信実装部_Skel94、95、メッセージ空間通信実装部_Stub96、97を生成する。

【0196】次に、アプリ側通信実装部56、57とメッセージ空間通信実装部58、59の各々を、メッセージ空間に依存するネームサーバであるrmiregistryサーバ98に登録する。ここで、メッセージ空間通信実装部_Stub96は、送信側のJavaVMで利用されるメッセージ空間通信実装部58の分身である。このオブジェクトにより、アプリ側通信実装部56がメッセージ空間3のメッセージ空間通信実装部58のメソッドを呼び出す際に、ネットワークの存在を意識せずに直接メソッドを呼び出す感覚で実行できる。

【0197】また、アプリ側通信実装部56は、rmiregistryサーバ98に問い合わせることにより、通信相手であるメッセージ空間通信実装部58を探すことが可能である。

【0198】一方、メッセージ空間通信実装部_Skel94は、メッセージ空間のJavaVMに存在し、送信側のメッセージ空間通信実装部_Stub96からの通信を受け付ける。

【0199】このように、JavaRMIでは、StubとSkelがそれぞれネットワーク接続の出入口として機能することで、アプリ側通信実装部56とメッセージ空間通信実装部58の分散通信を実現する。アプリ側通信実装部_Skel90、91とアプリ側通信実装部_Stub92、93とメッセージ空間通信実装部_Skel95とメッセージ空間通信実装部_Stub97についても同様である。

【0200】次に、メッセージ空間インタフェース47(23、24)が持つインタフェース定義をどのように実現するかについて具体的に発明する。ただし、「メッセージハンドラのためのメソッド(E)」については、分散通信とは直接関係しないので、説明を省略する。なお、分散通信においてメソッドの引数として渡されるメッセージハンドラ36、送達確認用メッセージハンドラ35、メッセージ32及びコンテキスト34は、インタ

フェース `java.io.Serializable` を実装する必要がある。

【0201】メッセージ空間インタフェース 47 は、メソッド定義の集合であり、その実装は定義しない。この中身は、送信側管理部 25 と受信側管理部 26 でそれぞれ実装される。

【0202】送信側管理部 25 は、メッセージ空間インタフェース 23 が受けたメッセージを、アプリ側通信実装部 56 に素通しするだけなので、説明は省略する。また、受信側管理部 26 も、メッセージ空間インタフェース 24 が受けたメッセージをアプリ側通信実装部 57 に素通しするだけなので、説明は省略する。

【0203】次に、アプリ側通信実装部 56、57、メッセージ空間通信実装部 58、59 について説明する。

【0204】アプリ側通信実装部クラスは、メッセージ空間通信実装部オブジェクトを属性として持つ。コンストラクタが起動された時には、`rmiregistry` 98 に登録されているメッセージ空間通信実装部 58 を探し、属性にセットする。見つけれなかった場合はエラー処理を行う。なお、メッセージ空間通信実装部 58、59 は、予め `rmiregistry` 98 に登録されているものとする。

【0205】メッセージ空間通信実装部クラスは必ず `java.rmi.server.Unicast Remote Object` を継承し、またメッセージ空間管理部クラスを属性として持つ。

【0206】コンストラクタは `java.rmi.Remote Exception` の例外を必ずスローし、起動された時にはスーパークラスのコンストラクタを起動して、引数として渡されたメッセージ空間管理部オブジェクトを属性にセットする。

【0207】アプリ側通信実装部 56、57 のメソッドでは、実行の際に例外が発生しうるので、必ず `try-catch` 文を用いて例外をキャッチする。また、メッセージ空間通信実装部 58、59 のメソッドでは、`java.rmi.Remote Exception` の例外を必ずスローするように定義する。

【0208】上記第 1 の実施の形態で述べた個々のインタフェースについて、アプリ側通信実装部 56、57 と、メッセージ空間通信実装部 58、59 がどのように動作するかを、以下に説明する。

【0209】利用開始のためのメソッド (A) :

(1) 送信するメッセージ 5 のメッセージ空間 3 におけるコンテキストの取得

まずアプリ側通信実装部 56 において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部 58 側のメソッドが呼び出される。

【0210】次にメッセージ空間通信実装部 58 において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部 56 に返す。エラーが起きた場合は、`null` を返す。

【0211】(2) 送信するメッセージ 5 の種類を登録す

るメソッド (その 1)

まずアプリ側通信実装部 56 において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部 58 側のメソッドが呼び出される。

【0212】次にメッセージ空間通信実装部 58 において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部 56 に返す。エラーが起きた場合は、`null` を返す。

【0213】(3) 送信するメッセージ 5 の種類を登録するメソッド (その 2)

まずアプリ側通信実装部 56 において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部 58 側のメソッドが呼び出される。

【0214】次にメッセージ空間通信実装部 58 において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部 56 に返す。エラーが起きた場合は、`null` を返す。

【0215】(4) 送達確認を `push` 形式で受け取るための確認用ハンドラを登録するメソッド

まずアプリ側通信実装部 56 において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部 58 側のメソッドが呼び出される。

【0216】次にメッセージ空間通信実装部 58 において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部 56 に返す。エラーが起きた場合は、`null` を返す。

【0217】(5) 受信するメッセージ 5 の種類を登録するメソッド

まずアプリ側通信実装部 57 において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部 59 側のメソッドが呼び出される。

【0218】次にメッセージ空間通信実装部 59 において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部 57 に返す。エラーが起きた場合は、`null` を返す。

【0219】(6) メッセージ 5 を `push` 形式で受け取るためのメッセージハンドラを登録するメソッド

まずアプリ側通信実装部 57 において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部 59 側のメソッドが呼び出される。

【0220】次にメッセージ空間通信実装部 59 において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部 57 に返す。エラーが起きた場合は、`null` を返す。

【0221】(7) メッセージ 5 の不着検出のための、属性を設定するメソッド

アプリ側通信実装部 57 において、所定のメソッドが実行される。この中で、スケジュールの設定を行う。メッセージ通信は行わない。

【0222】メッセージ送信のためのメソッド (B) :

(1)メッセージ5を送信するためのメソッド

まずアプリ側通信実装部56において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部58側のメソッドが呼び出される。

【0223】次にメッセージ空間通信実装部58において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部56に返す。エラーが起きた場合は、nullを返す。

【0224】(2)送達確認を送信するためのメソッド
まずアプリ側通信実装部57において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部59側のメソッドが呼び出される。

【0225】次にメッセージ空間通信実装部59において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部57に返す。エラーが起きた場合は、nullを返す。

【0226】(3)送達確認をpull形式で受信するためのメソッド

まずアプリ側通信実装部56において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部58側のメソッドが呼び出される。

【0227】次にメッセージ空間通信実装部58において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部56に返す。エラーが起きた場合は、nullを返す。

【0228】(4)メッセージ5をpull形式で受信するためのメソッド

まずアプリ側通信実装部57において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部59側のメソッドが呼び出される。

【0229】次にメッセージ空間通信実装部59において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部57に返す。エラーが起きた場合は、nullを返す。

【0230】メッセージ検索のためのメソッド(C)：

(1)メッセージ空間3でやり取りされているメッセージ5の種類の一覧を返すメソッド

まずアプリ側通信実装部56、57において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部58、59側のメソッドが呼び出される。

【0231】次にメッセージ空間通信実装部58、59において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部56、57に返す。エラーが起きた場合は、nullを返す。

【0232】(2)指定されたメッセージ5の送信オブジェクト一覧を返すメソッド

まずアプリ側通信実装部56、57において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部58、59側のメソッドが呼び出される。

【0233】次にメッセージ空間通信実装部58、59

において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部56、57に返す。エラーが起きた場合は、nullを返す。

【0234】(3)指定されたメッセージ5の受信オブジェクト一覧を返すメソッド

まずアプリ側通信実装部56、57において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部58、59側のメソッドが呼び出される。

【0235】次にメッセージ空間通信実装部58、59において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部56、57に返す。エラーが起きた場合は、nullを返す。

【0236】(4)指定されたメッセージ5の履歴一覧を返すメソッド

まずアプリ側通信実装部56、57において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部58、59側のメソッドが呼び出される。

【0237】次にメッセージ空間通信実装部58、59において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部56、57に返す。エラーが起きた場合は、nullを返す。

【0238】(5)指定されたメッセージ5が指定された時間以降にやり取りされているかどうかを調べるためのメソッド

まずアプリ側通信実装部56、57において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部58、59側のメソッドが呼び出される。

【0239】次にメッセージ空間通信実装部58、59において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部56、57に返す。エラーが起きた場合は、nullを返す。

【0240】利用終了のためのメソッド(D)：

(1)メッセージ空間3からの離脱を実施するメソッド

まずアプリ側通信実装部56、57において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部58、59側のメソッドが呼び出される。

【0241】次にメッセージ空間通信実装部58、59において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部56、57に返す。エラーが起きた場合は、nullを返す。

【0242】以上のように、本実施の形態においては、既存の分散システム間通信の実装技術として代表的なJavaRMIのための参照モデルのマッピング手法を示した。これにより、JavaRMIを利用したシステムの開発が容易になる。また、上記第1の実施の形態で説明したメッセージングシステムと同様のモデルを利用することで、例えばCORBAやJava及びECJなどのような他の分散システム間通信の実装技術との間で内部構造に互換性を持たせることもできる。したがって、ソフトウェア部品の共通化／再利用や保守／管理が容易になり、仕様変更の波及範

図を明確化できる。

【0243】(第4の実施の形態)本実施の形態においては、ECJのための参照モデルのマッピング手法について説明する。

【0244】図31は、分散通信技術にECJを使用した場合の模式図である。この図31は、上記図26中における分散通信技術に依存する部分をECJで実現している。分散通信技術に依存しない部分についての具体的な説明を省略する。

【0245】ECJの主な構成と動作を図32に示す。ECJは、イベントID属性とイベントデータ属性を持つイベントオブジェクト99、イベントIDに対応する処理を行うハンドラ100、イベントIDとそれに対応するイベントのハンドラ100の対応表101を持つLocal Manager 102から構成される。外部からイベント99を受け付けたLocal Manager 102は、自分が持つ対応表101に照らしあわせて、該当するハンドラ100を起動する。アプリケーション開発者は、イベント99に底じたハンドラ100を作成する。

【0246】ECJはイベント配送型であるため、分散通信を行うに際して、あるアクションに対する返り値を取得する方法がない。そこで、図33に示すWakeup方式を用いることにより、RMIと同様の操作を実現する。なおWakeup方式では、ECJのLocal Managerクラスを拡張し、ロックインスタンスとイベントオブジェクトの管理を実施可能としたECLMS Local Managerクラス、及びECJのUser Eventクラスを拡張し、イベントID属性とイベントデータ属性に加えて返り値イベントID属性を持つクラスECJ Message Eventクラスを定義して用いる。

【0247】これにより、例えば、イベントID0001と、引数となるイベントデータと、返り値イベントID1000を持つイベントを発生させることにより、Wakeup Handlerによって、イベントID1000と、返り値となるイベントデータを持つイベント353を確実に発生させ、返り値を取得することが可能になる以下にWakeup方式の流れ(上記図31、図33)について説明する。例として、ID0001のイベントを発生させて、返り値としてID1000のイベントが発生するものとする。

(1)送信側54のアプリ側通信実装部56は、イベントID1000とWakeup Handler 103を、Local Manager 104に登録する。さらに、ID1000のロックインスタンスを生成し、Local Manager 104に登録する。

【0248】(2)イベントIDが0001、イベントデータが引数となるデータ、返り値イベントIDが1000であるイベント105を、IPマルチキャストを通じてネットワーク106に配信する。

【0249】(3)送信側54のアプリ側通信実装部56は、IDが1000であるロックインスタンスが存在している間はsleepする。

【0250】(4)メッセージ空間3のLocal Manager 10

7に、イベントID0001のイベント105が配送される。

【0251】(5)Local Manager 107は、イベントID0001に対応するハンドラを起動する。起動されたハンドラは何らかの処理を実行したのち、イベントID1000のイベントを生成して、IPマルチキャストを通じてネットワーク106に配信する。

【0252】(6)送信側54のアプリ側通信実装部56のLocal Manager 104に、イベントID1000のイベントが配送される。

【0253】(7)Local Manager 104は、イベントID1000に対応するWakeup Handler 103を起動する。

【0254】この中でWakeup Handler 103は、Local Manager 104にイベントID1000のイベントを保存し、イベントIDと同じIDのロックインスタンスを消去する。

【0255】(8)送信側54のアプリ側通信実装部56は、上記(3)のsleepからぬけて、処理を再開する。Local Manager 107が保持するイベントID1000のイベントのイベントデータをもとに処理を進める。

【0256】(9)イベントID1000のイベントを削除し、イベント登録を解除する。

【0257】分散通信を実施するために、アプリケーション開発者は以下の操作を行う。

【0258】まず、ECJのクラスを用いて、Local Manager 104、107～109、Wakeup Handler 103、110～112、イベント105、113～115を生成する。その後、各Local Managerに対して、イベントIDとアプリ側通信実装部56、57、メッセージ空間通信実装部58、59を登録する。ここでは、アプリ側通信実装部56、57、メッセージ空間通信実装部58、59がハンドラとしての役割を持つ。このように、ECJを用いた場合は、ECJ本来の機能に加えてWakeup方式を用いることで、アプリ側通信実装部とメッセージ空間通信実装部の分散通信を実現する。

【0259】次に、メッセージ空間インタフェース47が持つ上記第1の実施の形態で述べたインタフェース定義をどのように実現するかについて具体的に説明する。ただし、「メッセージハンドラのためのメソッド(E)」については、分散通信とは直接関係しないので、説明を省略する。

【0260】主な通信は上記で述べたWakeup方式を用いる。また、ECJの機能で実現できる部分に関しては、特に明示的な通信を行わない場合もある。上記図26におけるメッセージ空間インタフェース23、24、送信側管理部25、受信側管理部36は、RMIマッピングにおけるものと同じであるので説明を省略する。

【0261】次に、アプリ側通信実装部56、57、メッセージ空間通信実装部58、59について説明する。

【0262】アプリ側通信実装部クラスは、Local Manager 104、108とトランスポートアダプタを属性として持つ。トランスポートアダプタは、IPマルチキャスト

を実現するうえで必要になる。コンストラクタが起動された時には、新しいLocalManager 104、108が生成され、属性にセットされる。また、新しいターミナルアダプタが生成され、属性にセットされる。生成に失敗した場合はエラー処理を行なう。

【0263】メッセージ空間通信実装部58、59のクラス定義は、アプリ側通信実装部クラスと同じなので、説明を省略する。また、メッセージ空間通信実装部クラスはハンドラとして振る舞うので、executeメソッドをオーバーライドする。executeメソッドの中で、switch-case文を用いることにより、イベントに対する適切なメソッドを起動することが可能である。なお、switch-case文に代えてメッセージ空間通信実装部クラスをサブクラス化して実現することも可能である。

【0264】アプリ側通信実装部56、57と、メッセージ空間通信実装部58、59の双方において、イベントを送信する時は、例外が発生しうるので、try-catch文を用いて、例外をキャッチする。

【0265】発生時の構成の項で述べた個々のインタフェースについて、アプリ側通信実装部56、57と、メッセージ空間通信実装部58、59がどのように動作するかを以下に説明する。

【0266】利用開始のためのメソッド(A)：

(1)送信するメッセージ5のメッセージ空間3におけるコンテキストの取得

まずアプリ側通信実装部56において、所定のメソッドがWakeUp方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部58側のメソッドが起動される。

【0267】次にメッセージ空間通信実装部58において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部56においてWakeUpHandlerが実行される。

【0268】その後、アプリ側通信実装部56において実行された所定のメソッドは処理を再開し、結果を返す。エラーが起きた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【0269】(2)送信するメッセージ8の種類を登録するメソッド(その1)

まずアプリ側通信実装部56において、所定のメソッドがWakeUp方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部58側のメソッドが起動される。次にメッセージ空間通信実装部58において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部56においてWakeUpHandlerが実行される。

【0270】その後、アプリ側通信実装部56において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイム

アウトなどの処理は作成するアプリケーションに依存する。

【0271】(3)送信するメッセージ8の種類を登録するメソッド(その2)

まずアプリ側通信実装部56において、所定のメソッドがWakeUp方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部58側のメソッドが起動される。

【0272】次にメッセージ空間通信実装部58において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部56においてWakeUpHandlerが実行される。その後、アプリ側通信実装部56において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【0273】(4)送達確認をpush形式で受け取るための確認用ハンドラを登録するメソッド

アプリ側通信実装部56において、所定のメソッドが実行される。この中で、アプリ側のLocal Manager 104にハンドラが登録される。

【0274】(5)受信するメッセージ5の種類を登録するメソッド

まずアプリ側通信実装部57において、所定のメソッドがWakeUp方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部59側のメソッドが起動される。

【0275】次にメッセージ空間通信実装部132において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部57においてWakeUpHandlerが実行される。

【0276】その後、アプリ側通信実装部57において実行された所定のメソッドは処理を再開し、結果を返す。エラーが起きた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【0277】(6)メッセージ5をpush形式で受け取るためのメッセージハンドラを登録するメソッド

アプリ側通信実装部57において、所定のメソッドが実行される。この中で、アプリ側のLocal Manager 108にハンドラが登録される。

【0278】(7)メッセージ5の不着検出のための、属性を設定するメソッド

アプリ側通信実装部57において、所定のメソッドが実行される。この中で、アプリ側のアプリケーションに対してスケジュールが登録される。

【0279】メッセージ送信のためのメソッド(B)：

(1)メッセージ5を送信するためのメソッド

アプリ側通信実装部56において、所定のメソッドが実行される。この中で、メッセージをイベントとして配送

する。

【0280】(2)送達確認を送信するためのメソッド
アプリ側通信実装部57において、所定のメソッドが実行される。この中で、メッセージをイベントとして配送する。

【0281】(3)送達確認をpull形式で受信するためのメソッド
まずアプリ側通信実装部56において、所定のメソッドがWakeUp方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部58側のメソッドが起動される。

【0282】次にメッセージ空間通信実装部58において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部56においてWakeUpHandlerが実行される。

【0283】その後、アプリ側通信実装部56において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【0284】(4)メッセージ5をpull形式で受信するためのメソッド
まずアプリ側通信実装部57において、所定のメソッドがWakeUp方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部59側のメソッドが起動される。

【0285】次にメッセージ空間通信実装部59において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部57において、WakeUp Handlerが実行される。

【0286】その後、アプリ側通信実装部57において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【0287】メッセージ検索のためのメソッド(C)：
(1)メッセージ空間3でやり取りされているメッセージ5の種類の一覧を返すメソッド
まずアプリ側通信実装部56、57において、所定のメソッドがWakeUp方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部58、59側のメソッドが起動される。

【0288】次にメッセージ空間通信実装部58、59において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部56、57においてWakeUp Handlerが実行される。

【0289】その後、アプリ側通信実装部56、57において実行された所定のメソッドが処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依

存する。

【0290】(2)指定されたメッセージ8の送信オブジェクト一覧を返すメソッド
まずアプリ側通信実装部56、57において、所定のメソッドがWakeUp方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部58、59側のメソッドが起動される。

【0291】次にメッセージ空間通信実装部58、59において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部56、57においてWakeUp Handlerが実行される。

【0292】その後、アプリ側通信実装部56、57において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【0293】(3)指定されたメッセージ8の受信オブジェクト一覧を返すメソッド
まずアプリ側通信実装部56、57において、所定のメソッドがWakeUp方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部58、59側のメソッドが起動される。

【0294】次にメッセージ空間通信実装部58、59において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部56、57においてWakeUp Handlerが実行される。

【0295】その後、アプリ側通信実装部56、57において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【0296】(4)指定されたメッセージ5の履歴一覧を返すメソッド
まずアプリ側通信実装部56、57において、所定のメソッドがWakeUp方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部58、59側のメソッドが起動される。

【0297】次にメッセージ空間通信実装部58、59において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部56、57においてWakeUp Handlerが実行される。

【0298】その後、アプリ側通信実装部56、57において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【0299】(5)指定されたメッセージ5が指定された時間以降のやり取りされているかどうかを調べるためのメソッド
まずアプリ側通信実装部56、57において、所定のメ

ソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部58、59側のメソッドが起動される。

【0300】次にメッセージ空間通信実装部58、59において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部56、57においてWakeup Handlerが実行される。

【0301】その後、アプリ側通信実装部56、57において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【0302】利用終了のためのメソッド(D)：

(1)メッセージ空間3からの離脱を実施するメソッド
アプリ側通信実装部56、57において、所定のメソッドが実行される。この中で、アプリ側のLocal Manager 104、108に対してメッセージハンドラを登録解除する。

【0303】以上のように、本実施の形態においては、既存の分散システム間通信の実装技術として代表的なECJのための参照モデルのマッピング手法を示した。これにより、ECJを利用したシステムの開発が容易になる。また、上記第1の実施の形態で説明したメッセージングシステムと同様のモデルを利用することで、例えばCORBAやJavaRMIなどのような他の分散システム間通信の実装技術との間で内部構造に互換性を持たせることもできる。したがって、ソフトウェア部品の共通化／再利用や保守／管理が容易になり、仕様変更の波及範囲を明確化できる。

【0304】

【発明の効果】以上詳記したように、本発明を利用してメッセージ交換の実装のための手段や仕組みを統一化することで、統一的なメッセージ交換のモデルを与えることができる。

【0305】これにより、実装技術が異なる場合であっても内部の構造に互換性を持たせることができる。したがって、ソフトウェアの共通化／再利用、保守／管理が容易となり、仕様を変更する場合に影響を受ける範囲を明確化できる。

【0306】また、本発明を利用して分散システムにおけるメッセージ交換のためのインタフェースやプロトコルを共通化することにより、実装基盤に依存しない統一的なメッセージングの仕組みを提供できる。

【0307】また、本発明ではメッセージ交換のための構造を階層型としているため、業務の細部の仕様変更、新しい業務の追加、ハードウェア構成の変更など、各種仕様変更に対応できる。また、例えばメッセージの送達確認機能、メッセージ空間に対する検索機能、メッセージ不着信検出及びその対処機能を容易に実現させることができる。

【図面の簡単な説明】

【図1】本発明の第1の実施の形態に係るメッセージングシステムを利用する分散システムの構成を例示するブロック図。

【図2】チャンネルマネージャの派生を示す図。

【図3】チャンネルの派生を示す図。

【図4】キューチャンネルによるメッセージ伝達の様子を示す図。

【図5】ストアチャンネルによるメッセージ伝達の様子を示す図。

【図6】シンプルチャンネルによるメッセージ伝達の様子を示す図。

【図7】メッセージプールの派生を示す図。

【図8】メッセージ空間を利用する場合の流れ図。

【図9】メッセージ送信オブジェクトからみた初期化の流れ図。

【図10】メッセージ送信オブジェクトからみた運用の流れ図。

【図11】メッセージ受信オブジェクトからみた初期化の流れ図。

【図12】メッセージ受信オブジェクトからみた運用の流れ図。

【図13】同実施の形態に係るメッセージングシステムの仕組みを実現するための参照モデルを例示するクラス図。

【図14】チャンネルマネージャクラスのサブクラス化を示す図。

【図15】チャンネルクラスのサブクラス化を示す図。

【図16】メッセージプールのサブクラス化を示す図。

【図17】メッセージ送信部による初期化のシーケンス図。

【図18】メッセージ受信部による初期化のシーケンス図。

【図19】メッセージ送信部によるメッセージ送信のシーケンス図。

【図20】メッセージ受信部によるpush型のメッセージ受信のシーケンス図。

【図21】メッセージ受信部によるpull型のメッセージ受信のシーケンス図。

【図22】メッセージ送信部による終了処理のシーケンス図。

【図23】エラーの発生箇所／種類とその対処を担当する部分の関係を例示する図。

【図24】同実施の形態に係るメッセージングシステムによるキューを用いたメッセージ交換の一例を示す図。

【図25】キューチャンネルで記録される情報を示した図である。

【図26】同実施の形態に係るメッセージングシステムのオブジェクト図。

【図27】本発明のメッセージ空間の機能を分散させた

場合のオブジェクト図である。

【図28】CORBAを使用した場合のオブジェクト図（Push型）。

【図29】CORBAを使用した場合のオブジェクト図（Pull型）。

【図30】RMIマッピングを示す図。

【図31】EJFマッピングを示す図。

【図32】EJF構成図。

【図33】Wakeup方式のシーケンス図。

【符号の説明】

1…メッセージ送信オブジェクト

2…メッセージ受信オブジェクト

3…メッセージ空間

4…メッセージハンドラ

5…メッセージ

6…メッセージ空間管理部

7…チャンネルマネージャ

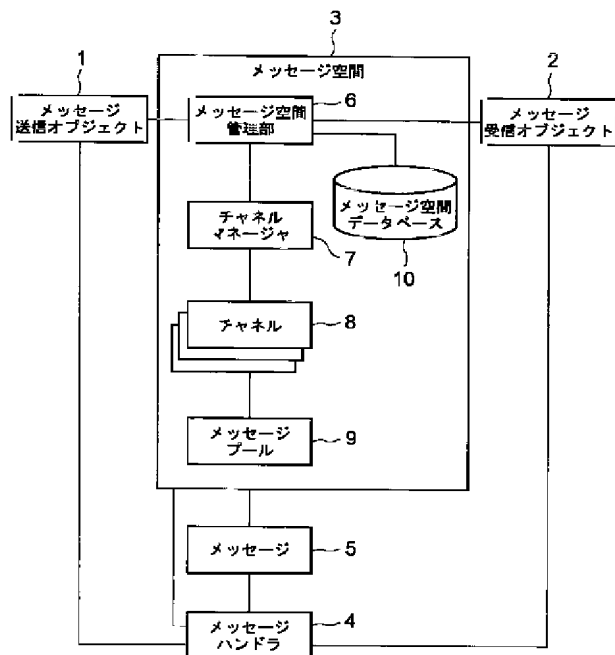
8…チャンネル

9…メッセージプール

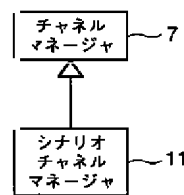
10…メッセージ空間データベース

47…メッセージ空間インタフェース

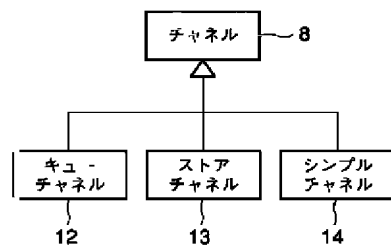
【図1】



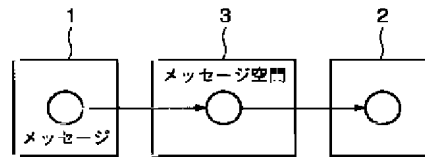
【図2】



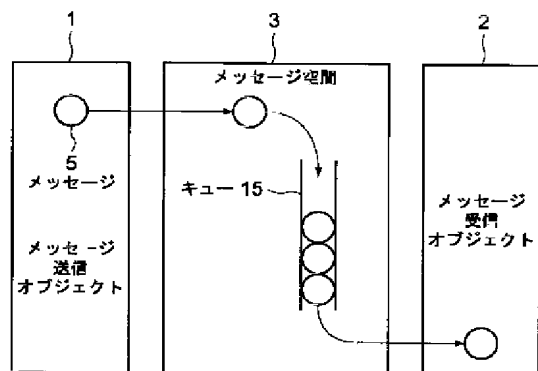
【図3】



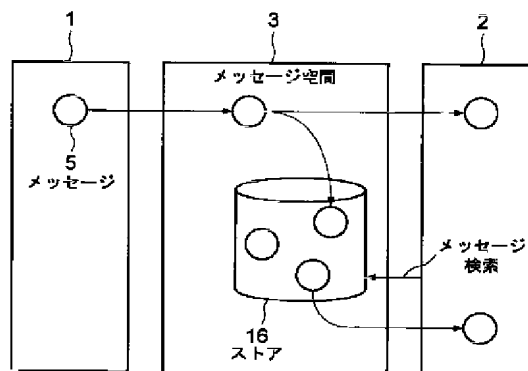
【図6】



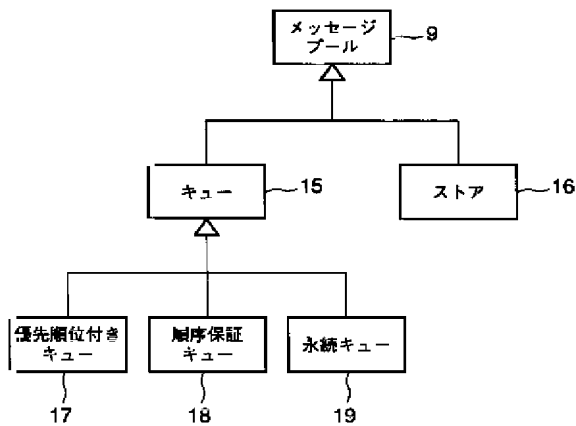
【図4】



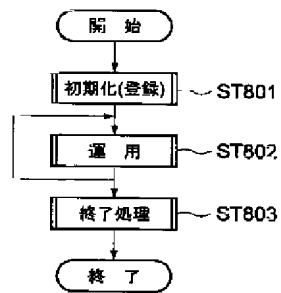
【図5】



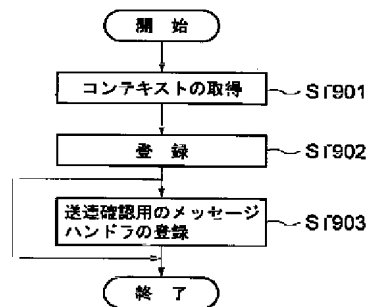
【図7】



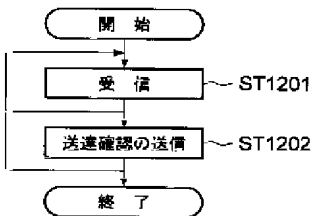
【図8】



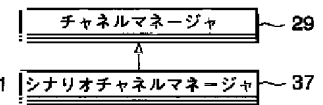
【図9】



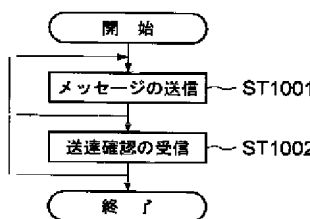
【図12】



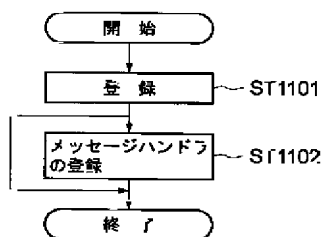
【図14】



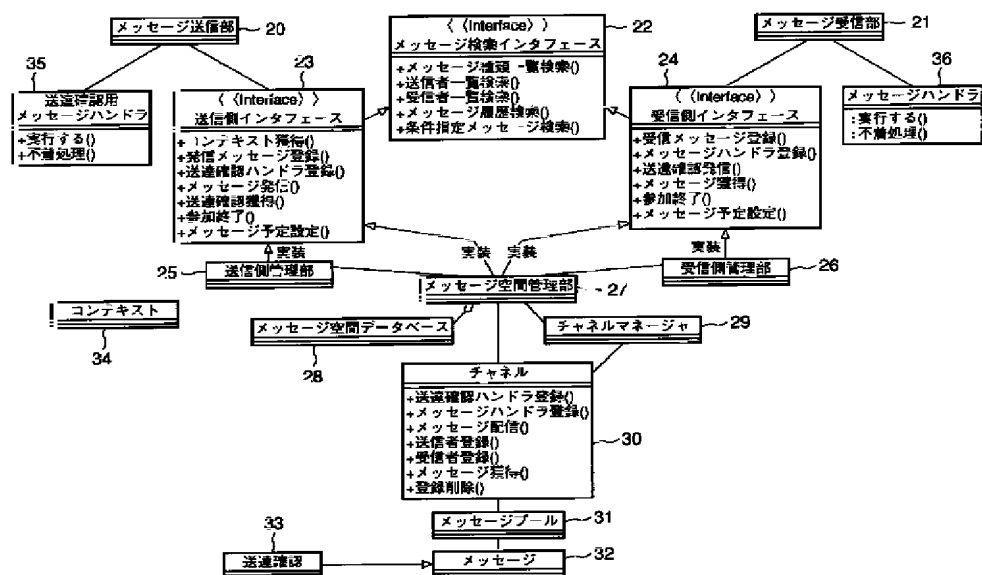
【図10】



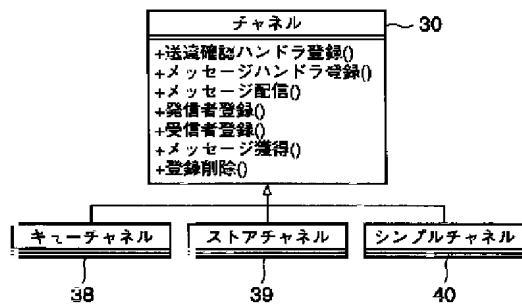
【図11】



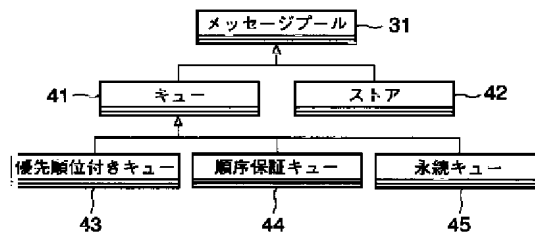
【図13】



【図15】



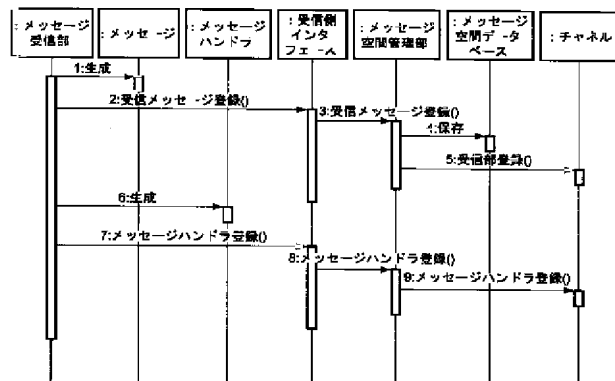
【図16】



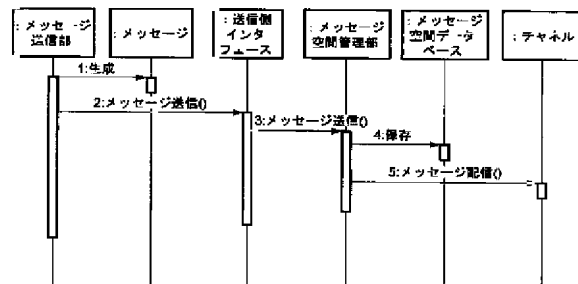
【図17】



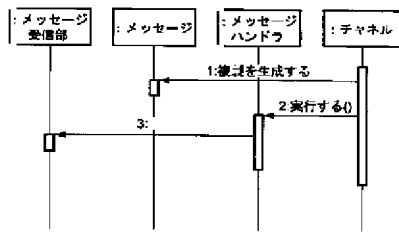
【図18】



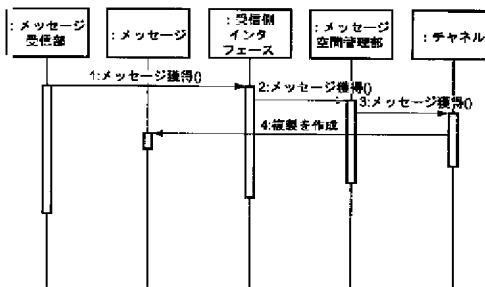
【図19】



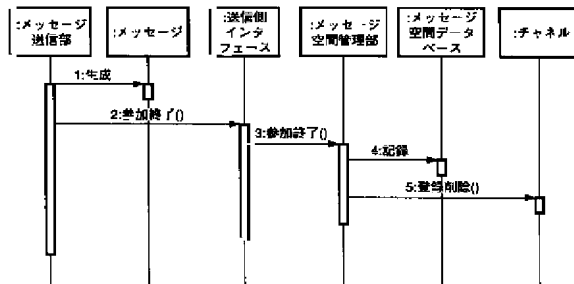
【図20】



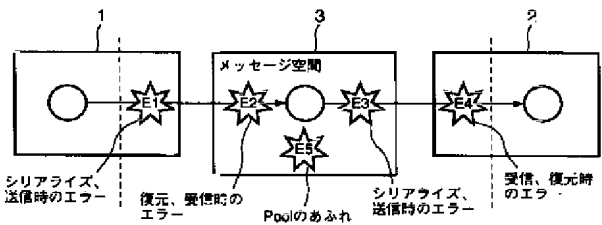
【図 2 1】



【图 22】



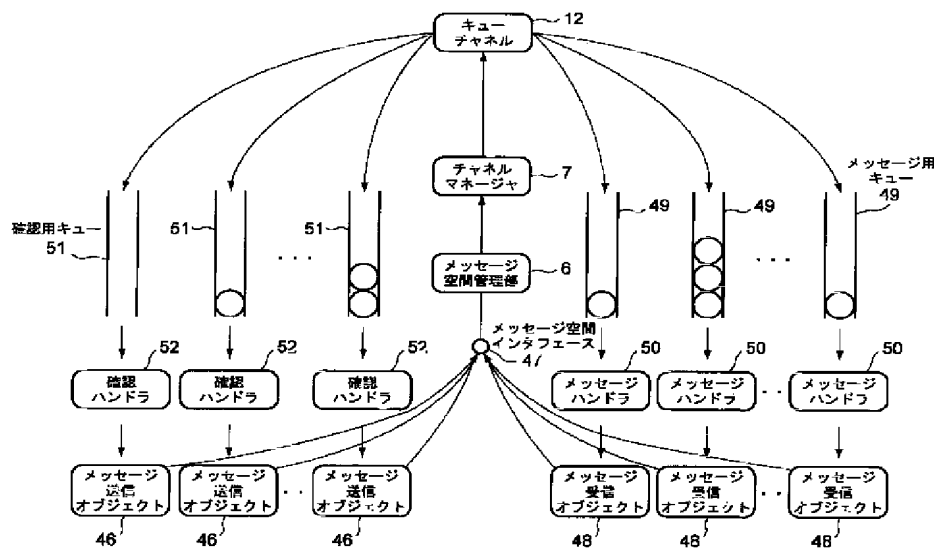
【图 23】



故障箇所とその対処担当者

- 1・送信側管理部
- 2・メッセージ空間管理部(送信側管理部へ再送要求)
- 3・メッセージ空間管理部
- 4・受信側管理部(メッセージ空間管理部へ再送要求)
- 5・メッセージ空間管理部

【例 24】

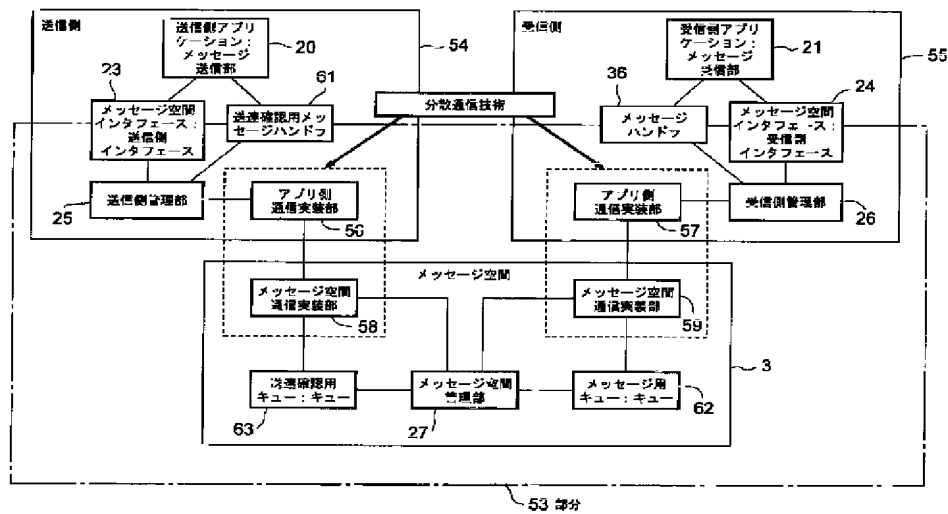


【图25】

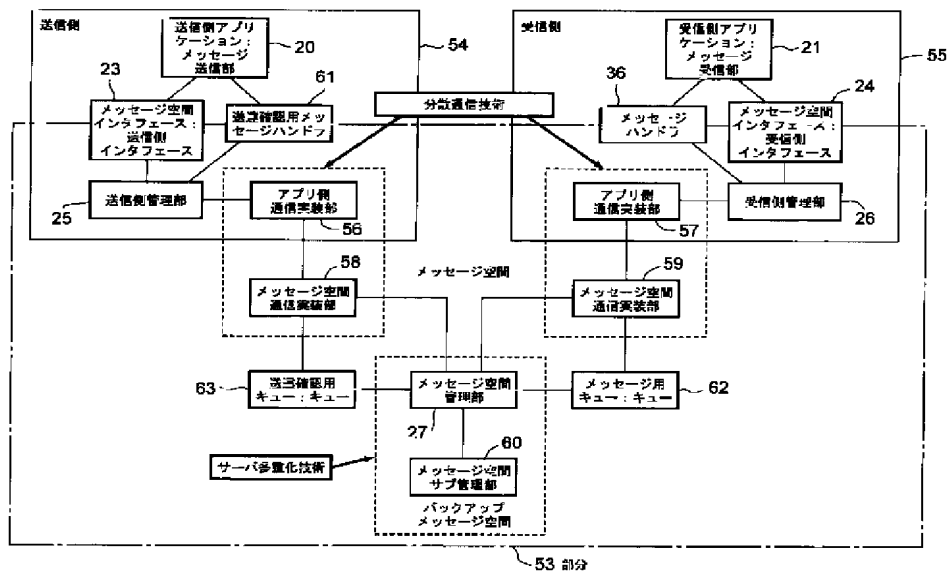
メッセージID	配信先1	配信先2	配信先3	...	情報
000001	確認済み	確認済み	確認済み	...	完了
000002	送信済み	確認済み	確認済み	...	未読
000003	送信済み	送信済み	送信済み	...	未読
⋮	⋮	⋮	⋮	⋮	⋮

キューチャネルで記録されるデータ

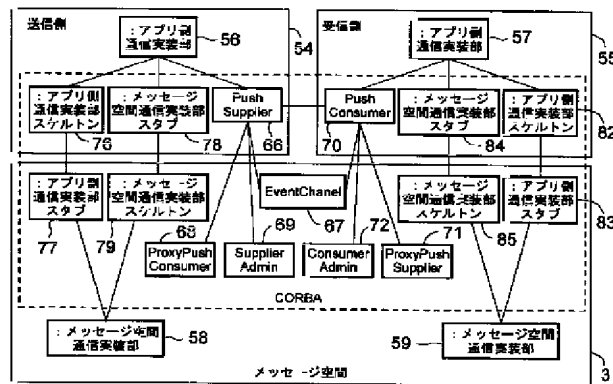
【图 26】



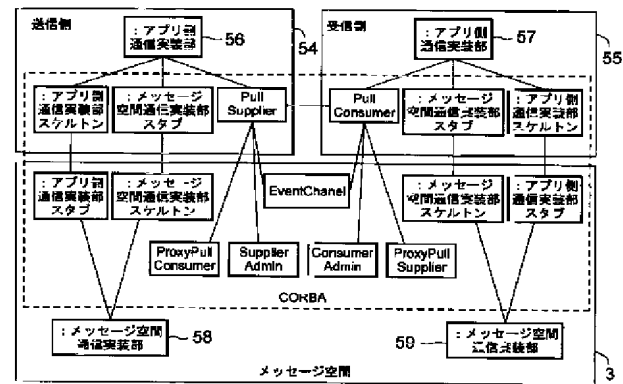
【图 27】



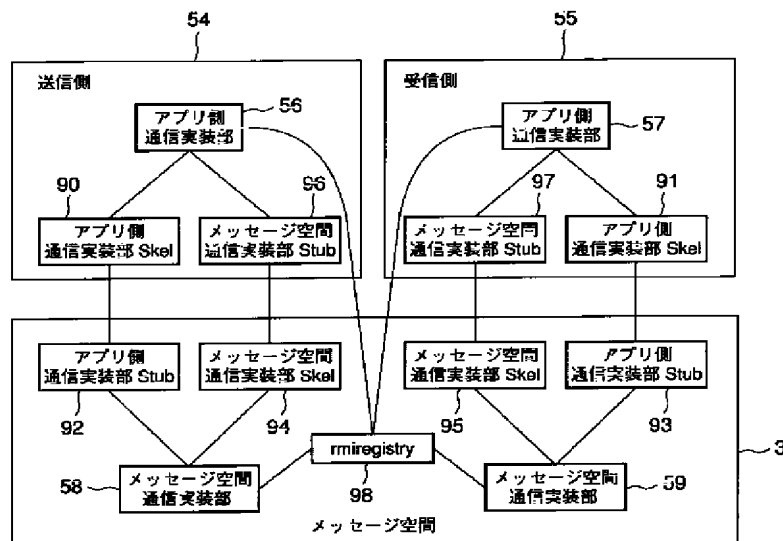
【図28】



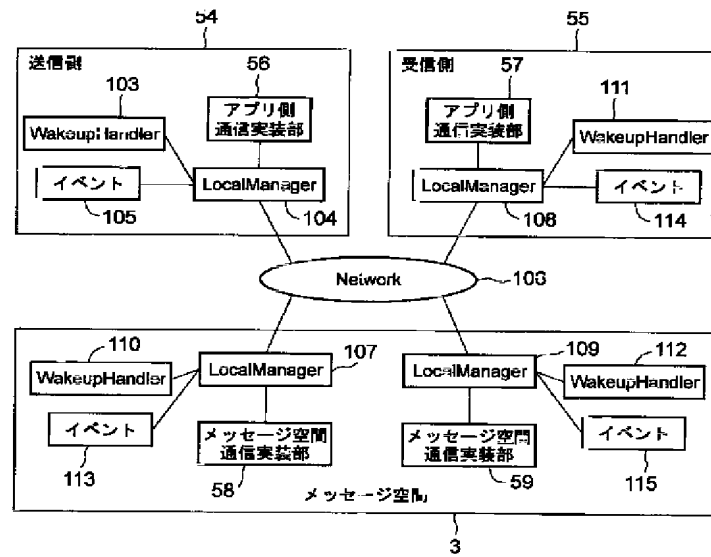
【図29】



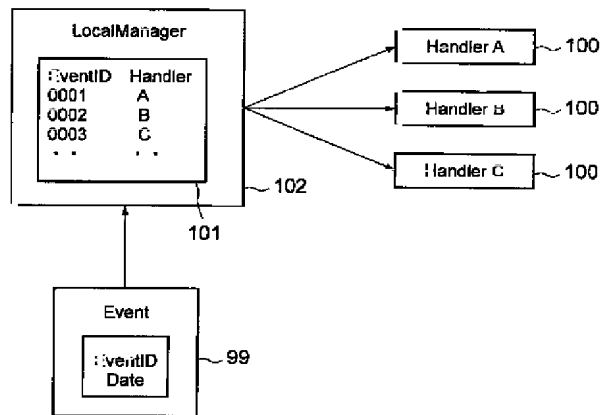
【図30】



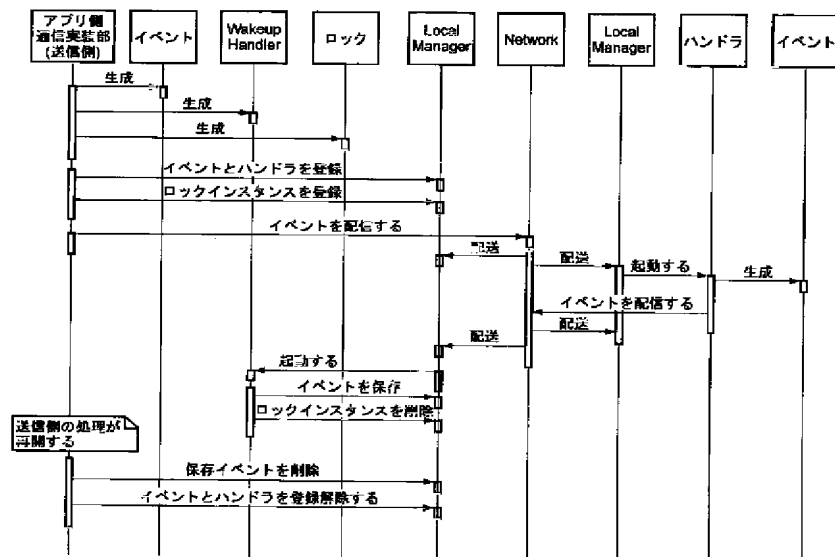
【図31】



【図32】



【図33】



フロントページの続き

(72)発明者 秋元 直人
東京都府中市東芝町1番地 株式会社東芝
府中事業所内

(72)発明者 鞍田 友美
東京都府中市東芝町1番地 株式会社東芝
府中事業所内

Fターム(参考) 5B045 BB31 GG01
5B089 GB01 GB10 JA11 KA13 KB06
KB09 KE01